Distance Consistent Labellings and the Local List Number

Department of Mathematics, Linköping University

Anders Henricsson

LiTH-MAT-EX–2023/15–SE

Credits: **16 hp**

Level: **G2**

Supervisor: **Carl Johan Casselgren**,
Department of Mathematics, Linköping University

Examiner: **Jan Snellman**,
Department of Mathematics, Linköping University

Linköping: **December 2023**

Abstract

We study the local list number of graphs introduced by Lennerstad and Eriksson. A labelling of a graph on n vertices is a bijection from vertex set to the set $\{1, \ldots, n\}$. Given such a labelling c a vertex u is distance consistent if for all vertices v and w $|c(u) - c(v)| = |c(u) - c(w)| + 1$ implies $d(u, w) \leq d(u, v)$. A graph G is k-distance consistent if there is a labelling with k distance-consistent vertices. The local list number of a graph G is the largest k such that G is k-distance consistent. We determine the local list number of cycles, complete bipartite graphs and some trees as well as prove bounds for some families of trees. We show that the local list number of even cycles is two, and of odd cycles is three. We also show that, if $k, l \geq 3$, the complete bipartite graph $K_{k,l}$ has local list number one, the star graph $S_n = K_{1,n}$ has local list number 3, and $K_{2,k}$ has local list number 2. Finally, we show that for each $n \geq 3$ and each k such that $3 \leq k \leq n$ there is a tree with local list number k.

Keywords:
distance-consistence, graph labelling, graph distance

URL for electronic version:
`http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-199624`

Sammanfattning

Vi studerar det lokala listtalet introducerat av Lennerstad och Eriksson. En märkning av en graf på n hörn är en bijektion från hörnmängden till mängden $\{1, \ldots, n\}$. Givet en sådan märkning c är ett hörn u avståndskonsistent om för alla hörn v och w $|c(u) - c(v)| = |c(u) - c(w)| = 1$ implicerar $d(u, w) \leq d(u, v)$. En graf G är k-avståndskonsistent om det finns en märkning med k avståndskonsistenta hörn. Det lokala listtalet av en graf G är det största k sådan att G är k-avståndskonsistent. Vi bestämmer den lokala listtalet av cykler, kompletta bipartita grafer och vissa träd och visar som gränser för några familjer av träd. Vi visar att det lokla listtalet av jämna cykler är två, och av udda cykler är tre. Vi visar också att, om $k, l \geq 3$, den kompletta bipartita grafen $K_{k,l}$ har lokalt listtal ett, stjärngrafen $S_n = K_{1,n}$ har lokalt listtal 3, och $K_{2,k}$ har lokalt listtal 2. Slutligen, visar vi att för varje $n \geq 3$ och varje k sådant att $3 \leq k \leq n$ finns ett träd med lokalt listtal k.

Nyckelord:

avstånds-konsestens, grafmärkning, grafavstånd

URL för elektronisk version:

`http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-199624`

Acknowledgements

I would like to thank my supervisor Carl Johan Casselgren for providing me with his notes on distance-consistent labellings, which were the starting point for this bachelor thesis, and for helping me throughout this project.

Nomenclature

Unless otherwise stated all graphs are simple, undirected and connected.

G	Graph
$V(G)$	Vertex set of graph G
$E(G)$	Edge set of graph G
$G[A]$	Induced subgraph of G with vertex set A.
$d_G(v)$	The degree of $v \in V(G)$.
$\mathrm{diam}(G)$	Diameter (longest shortest path between any pair of vertices) of graph G
S_n	Star graph with n leaves
$S_{k,l}$	Double star with leaf sets of size k and l.
$K_{k,l}$	Complete bipartite graph with partite sets of size k and l.

Contents

1 Introduction **1**

2 Prerequisites **3**
2.1 Some definitions 3
2.2 Background 5

3 The Local List Number **7**
3.1 Some observations 7
3.2 Cycles 8
3.3 Trees 9
3.4 Bipartite Graphs 18

A Computer calculations **25**

Chapter 1

Introduction

One way of adding information to a graph is by labelling its vertices. In our context a vertex labelling c of a graph $G = (V(G), E(G))$ is a bijection $c : V(G) \to \{1, 2, \ldots, |V(G)|\}$. Several authors have studied vertex labellings that encode some information about the graph, see [6].

A vertex labeling c gives rise to a distance $c(u, v) = |c(u) - c(v)|$. This distance is called the **list distance** as the vertex labeling gives a way to list vertices and can be interpreted as a "distance" between lists of vertices. A natural question is how closely this label-distance can be made to agree with the usual distance $d(u, v)$ between vertices u and v in a graph, that is, the length of a shortest path between u and v. Lennerstad & Eriksson studied labellings which minimize $\sum_{u,v \in V(G)} (c(u, v) - d(u, v))^2$ in [8].

Clearly we have $d(u, v) = c(u, v)$ for all vertices u and v in a graph G if and only if G is a path. We also have $d(u, v) = c(u, v)$ for all vertices u and v if and only if $d(u_1, v_1) < d(u_2, v_2) \implies c(u_1, v_1) < c(u_2, v_2)$ for all vertices u_1, u_2, v_1, v_2. A slight relaxation of this criterion is that $d(u_1, v_1) < d(u_2, v_2) \implies c(u_1, v_1) \leq c(u_2, v_2)$ for all $u_1, u_2, v_1, v_2 \in V(G)$. We call this criterion the **list criterion** and graphs satisfying it **list graphs**. Lennerstad & Eriksson studied list graphs in [8] and found examples of list graphs for all $|V(G)|$ and $|E(G)|$ such that $|V(G)| - 1 \leq |E(G)| \leq \binom{|E(G)|}{2}$. They also established some properties of list graphs.

A further relaxation of the list criterion is the **local list criterion** which is that $|d(u, w) \leq d(u, v)|$ for all vertices $u, v, w \in V(G)$ satisfying $|c(u) - c(v)| = |c(u) - c(w)| + 1$. This criterion is the main focus of this bachelors thesis. Specifically we aim to answer the question: for a graph G let c be a labelling that maximizes the number of vertices u satisfying the local list criterion: $d(u, w) \leq$

$d(u,v) \forall v, w \in V(G)$ satisfying $|c(u) - c(v)| = |c(u) - c(w)| + 1$, what is the number of such vertices? We call this number the **local list number**.

We find that the local list number of even cycles is 2, and of odd cycles is 3. For $k, l \geq 3$ we show that the complete bipartite graph $K_{k,l}$ has local list number 1. Furthermore we show that $K_{2,k}$ has local list number 2 and $K_{1,n}$ (which is also the star graph S_n) has local list number 3. We show that all trees are 2-distance-consistent, and that for each $n \geq 3$ and k such that $3 \leq k \leq n$ there is a tree with local list number k. We show that the local list number of the double star $S_{k,l}$ is 3 if and only if $\max\{k, l\} \geq 3$. We also find the local list number of all trees on 9 or fewer vertices by computer calculation. We find one such tree with local list number 2.

Chapter 2

Prerequisites

2.1 Some definitions

The following are some basic definitions of graph theory, as well as some definitions of some less commonly used concepts that are pertinent to this thesis. For more information on basic graph theory see for example [5].

A **graph** G is a pair $(V(G), E(G))$ of sets where the elements of $E(G)$ are 2-element subsets of $V(G)$. The set $V(G)$ is called the **vertex set** and its elements **vertices**. $E(G)$ is called the **edge set** and its elements **edges**. We will usually write the edge $e = \{x, y\} \in E(G)$ as xy. Two vertices x and y are said to be **adjacent** if $xy \in E(G)$.

A graph H is a **subgraph** of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. If H contains all edges of G with both endpoints in $V(H)$ then H is an **induced subgraph** of G.

If H is a graph, $x \in V(G)$, $y \in V(G)$, and $e = xy$ then $G = H + e$ is the graph $(V(H), E(H) \cup e)$.

A **path** is an ordered list of vertices where each vertex in the list is adjacent to the vertex preceding it. The **path** P_n is a graph on the form $V(P_n) = \{x_1, \dots x_n\}, E(P_n) = \{x_1x_2, x_2x_3 \dots x_{n-1}x_n\}$. If u and v are vertices in on a path P uPv is the subpath of P with endpoints u and v.

A graph G is **connected** if for any pair of vertices x and y there is a path in G with endpoints x and y.

In this thesis we will assume all graphs to be connected.

The **distance** between two vertices u and v is the length of a shortest path between them and is denoted by $d(u, v)$. Two vertices u and v are called **neighbours** if $d(u, v) = 1$. The **degree** $d_G(v)$ of a vertex v is the number of neighbours of v.

Let $n \geq 3$ and let $P = v_1 \dots v_n$ be a path on n vertices. The graph $C_n :=$ $P + v_1 v_n$ is called a **cycle**. A graph which does not have a cycle as a subgraph is called **acyclic**, or a **forest**. A connected acyclic graph is called a **tree**. Vertices of trees of degree one are called **leaves** while vertices of degree greater than one are called **internal vertices**.

A graph G is **complete** if all possible edges are present, that is if $vu \in E(G)$ for all pairs of distinct vertices $(u, v) \in V(G)^2$. K_n denotes a complete graph on n vertices.

A graph G is called **bipartite** if there are two disjoint subsets X and Y of $V(G)$ such that $V(G) = X \cup Y$ and each edge has one end in X and one end in Y. X and Y are called **partite sets**. A bipartite graph G with partite sets X and Y is called **complete** if for all $x \in X$ and $y \in Y$ $xy \in E(G)$. $K_{n,m}$ denotes a complete bipartite graph with partite sets of size n and m. A **Star graph** S_n is a complete bipartite graph with one vertex in one partite set and n vertices in the other partite set.

A **vertex labeling** of a graph G is a bijective function $c : V(G) \to \{1, 2, \dots, |V(G)|\}$. We call the function $c(u, v) = |c(u) - c(v)|$ the **label distance** given by the labeling c.

We move on to some more uncommon definitions that have importance to this thesis. These concepts were introduced in [8], [7], and [4].

A graph G is a **list graph** if there is a vertex labeling c such that $\forall u_1, u_2, v_1, v_2 \in$ $V(G)$ $d(u_1, v_1) < d(u_2, v_2)$ implies $c(u_1, v_2) \leq c(u_2, v_2)$. Such a labeling is called **list-distance consistent** or **distance-consistent**. This condition is called the **weak list condition**.

The following definition was first introduced by Lennerstad in [7]. A graph G is a **local list graph** if there is a vertex labeling c such that $d(u, w) \leq d(u, v)$ holds for all $u, v, w \in V(G)$ satisfying $c(u, v) = c(u, w) + 1$. Such a labeling is called a **local list labeling**.

The following definitions are due to [4].

A vertex labeling c is k-**distance consistent** if there are k vertices u which satisfy $d(u, w) \leq d(u, v)$ for all vertices v, w satisfying $c(u, v) = c(u, w) + 1$. Given a labeling a vertex u is said to be **distance-consistent** if it satisfies this condition. A graph G is said to be k-**distance-consistent** if there is a k-distance-consistent labeling of G.

The **local list number** of G, denoted by $dc(G)$, is the largest k such that there is a k-distance consistent vertex labeling of G.

2.2 Background

Lennerstad and Eriksson introduced the concept of distance-consistence in [8]. Their approach of studying labellings where the list distance is close to the usual distance was novel. They hoped that this approach might help in describing structure for very large graphs and that it might find applications in graph theory and its applications.

A prior approach which is somewhat similar is to study adjacency labellings. These are "labellings" where it is possible to tell whether two vertices are adjacent only by their labels. Unlike our labellings the labels used in these schemes are binary codes and thus they are not labellings by our definition. These were studied in [9] where an algorithm for creating such a labelling using labels of $|V(G)|/2 + O(1)$ bits was shown. Prior to [9] results regarding the existence and length of particular kinds of such "labellings" were found in 1966 by M.A. Breuer [1]. A graph is said to be T-codable if there exists a particular kind of adjacency labelling where two vertices are adjacent iff the Hamming distance of their labels is less than or equal to T. In [2] Breuer shows the unexpected result that if a graph G is T'-codable for an even T' then G is T'-codable for all $T \geq T'$; however if T' is odd then G is not necessarily $(T'+1)$-codable.

Another approach is distance-labellings (also known as proximity-preserving labellings). Such a "labeling" allows one to compute the distance between any two vertices exactly. In [3] minimum lengths for such labellings of various classes of graphs are presented. These are (in general) not labellings by our definition as labels other than $1, 2, \ldots, V|(G)|$ are typically needed to achieve this.

In [8] Lennerstad and Eriksson studied labellings c that minimize $\sum_{u,v \in V(G)} (c(u,v) - d(u,v))^2$, as well as the weak list condition. They called such labellings distance-consistent. To avoid confusion we shall instead call such labellings **distance-minimizing**. They found an algorithm for finding such labellings.

Lennerstad and Eriksson proved the existence of list graphs with n vertices and k edges given that $n - 1 \leq k \leq n(n-1)/2$[8]. In addition they proved the following proprieties of list graphs (theorem 3.2 in [8]).

Theorem 2.2.1. *1. G is not a list graph if it contains a cycle graph C_n for any $n \geq 4$ as an induced subgraph.*

2. G is not a list graph if it contains a star graph S_n for any $n \geq 3$ as an induced subgraph.

3. G is not a list graph if it contains three vertices of degree one.

4. If G is a list graph and has two vertices u and v of degree one, then $c(u) = 1$ and $c(v) = n$, or $c(u) = n$ and $c(v) = 1$.

5. If G is a list graph, it has a Hamiltonian path. [8]

Note that condition 4. implies that G has at most two vertices of degree one: a property shared with path graphs. Condition 5. was proved by showing that the list labeling gives a hamlitonian path.

In [7] Lennerstad further proved that list labellings are distance-minimizing.

Lennerstad continued the research into distance-minimizing labellings in [7] using qualitative and quantitative characterizations.

In [7] local list graphs are first defined and some properties of local list graphs are proved. Casselgren characterized local list graphs in [4], and provided a proof sketch. Here we shall flesh out his proof, but first we prove a useful proposition:

Proposition 2.2.2. *Let c be a vertex labeling of a graph G. If $u \in V(G)$ is distance-consistent then u must have a neighbour labeled $c(u) - 1$ or $c(u) + 1$.*

Proof. Assume $u \in V(G)$ is distance-consistent but has no neighbour labeled $c(u) + 1$ or $c(u) - 1$. Let v be a neighbour of u and let $w_1 = c^{-1}(c(u) - 1)$ and $w_2 = c^{-1}(c(u) + 1)$, if they exist (at least one of w_1 and w_2 must exist if $|V(G)| > 1$). We either have $c(u, w_1) < c(u, v)$ or $c(u, w_2) < c(u, v)$, but $d(u, w_1) > d(u, v) = 1$ and $d(u, w_2) > d(u, v) = 1$, contradicting that u is distance-consistent under c.

Theorem 2.2.3. *A graph G is a local list graph iff it has a hamiltonian path P where if two vertices u and v are adjacent then the subgraph induced by the vertices on uPv is a clique. [4]*

Proof. Based on a proof sketch in [4]. Let c be a local list labeling of G. First we show that for any $i \in \{1, \ldots, |V(G)| - 1\}$ $u_i := c^{-1}(i)$ is adjacent to $u_{i+1} := c^{-1}(i + 1)$. To find a contradiction assume that u_i and u_{i+1} are not adjacent. Since G is connected there is a path P_i from u_i to u_{i+1}. Let w_i be the vertex after u_i on P_i. $c(w_i) \not\geq i + 2$ because $d(u_i, w_i) < d(u_i, u_{i+1})$. But we also have $c(w_i) \not\leq i - 1$ because $c(u_{i+1}, w_i) \geq 2 \geq c(u_{i+1}, u_i) = 1$ if $c(w_i) \leq i - 1$, but $d(u_{i+1}, w_i) < d(u_i, u_{i+1})$. Thus w cannot have any label, which is a contradiction.

Let $P = c^{-1}(1)c^{-1}(2) \ldots c^{-1}(|V(G)|)$. Note that P is hamiltonian. Let u and v be adjacent. The vertices on uPv are labeled $c(u), c(u) + 1, \ldots c(v)$. Take $w \in V(uPv)$. Since u is adjacent to v and $c(u, w) < c(u, v)$ w must also be adjacent to u. Similarly a vertex u' adjacent to u on uPv must be adjacent to all vertices in $u'Pv$. The same applies to a vertex adjacent to u' on $u'Pv$, and so on. Thus all possible edges in $G[uPv]$ are present, and therefore $G[uPv]$ is a clique. □

Chapter 3

The Local List Number

3.1 Some observations

Unlike the list criterion which is global and deals with vertex quadruples the local list criterion deals with vertex triples deals with vertex triples.

The following proposition gives an equivalent definition of distance-consistency of vertices.

Proposition 3.1.1. *Let c be a labelling of a graph G and take $u \in V(G)$. Then u is distance-consistent iff $c(u,w) < c(u,v) \implies d(u,w) \leq d(u,v)$ for all vertices v and w.*

Proof. Let u, v and w be vertices and let u be distance-consistent and $c(u,w) < c(u,v)$. If $c(u,v) = c(u,w)+1$ we must have $d(u,w) \leq d(u,v)$ because u is distance-consistent. Otherwise, if $c(u,v) - c(u,w) := n > 1$, create a list of vertices $w_1, w_2, \ldots w_n$ such that $c(u,w_i) = i$ for all $i \leq n$. For all such i we must have $d(u,w_i) \leq d(u,w_{i+1})$, as otherwise u, w_i and w_{i+1} would contradict the local list criterion. Thus $d(u,w) \leq d(u,v)$.

To show sufficiency assume $u \in V(G)$ is such that $c(u,w) < c(u,v) \implies d(u,w) \leq d(u,v)$ for all vertices v and w. u is distance-consistent as $c(u,v) = c(u,w)+1 \implies d(u,w) < d(u,v)$. □

The perspective on distance-consistency of vertices provided by this proposition will be of great use.

The following proposition was proved by Casselgren in [4].

Proposition 3.1.2. *All graphs are 1-distance-consistent. Moreover, for any vertex u there is a labelling such that u is distance-consistent.*

Proof. Pick a vertex u and label it 1. Let $v_1, \dots v_{|V(G)|}$ be such that $d(u, v_i) \leq d(u, v_j)$ if $i \leq j$ and let $c(v_i) = i$ for all i. u is distance-consistent with respect to c. □

Lemma 3.1.3. *Let u be a vertex which is adjacent to all other vertices in $V(G)$. Given any labelling c of G u is distance-consistent. Thus a non-complete graph with k such vertices is $(k+1)$-distance-consistent, as there is vertex v which is not adjacent to all other vertices and a labeling c such that v is distance-consistent with respect to c.*

Proof. Since $d(u, w) = 1 \leq d(u, v)$ for any vertex v, u must be distance-consistent. □

When checking if a vertex u is distance-consistent, that is if $d(u, w) \leq d(u, v)$ for all vertices v and w satisfying $c(u, v) = c(u, w) + 1$, it is not necessary to check the case $v = u$ since there is no w satisfying $0 = c(u, v) = c(u, w) + 1$. Neither is it necessary to check the case $w = u$ since $0 = d(u, w) \leq d(u, v)$ for all v.

Proposition 3.1.4. *If u is distance-consistent then one vertex at the maximum distance from u must be at maximum list distance from u, that is one vertex at maximum distances from u must be labelled 1 or $|V(G)|$.*

Proof. Let u be distance-consistent and let v be the vertex furthest from u. v must be labelled 1 or $|V(G)|$ as otherwise either $c(u, v) < c(u, c^{-1}(1))$ or $c(u, v) < c(u, c^{-1}(|V(G)|))$ despite $d(u, v) > d(u, c^{-1}(1))$ and $d(u, v) > d(u, c^{-1}(|V(G)|))$, which would be a contradiction to proposition 3.1.1.

3.2 Cycles

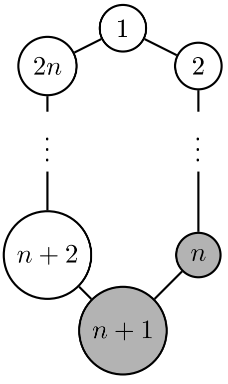
The case of cycles is simple in that $dc(C_n)$ only depends on the parity of n.

Theorem 3.2.1. $dc(C_{2n}) = 2$

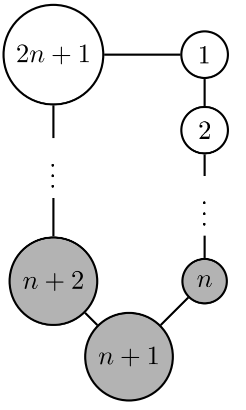
Proof. In an even cycle there is only one vertex at maximum distance from a given vertex w. Therefore, by Proposition 3.1.4 and the pigeonhole principle, there can be at most two distance-consistent vertices in any labelling of C_{2n}.

Let c be a labelling where the vertex labelled 1 is adjacent to vertex labelled 2, 2 is adjacent to 3 and so on (see figure 3.1a). Take $v \in V(G)$. For both $u_1 = c^{-1}(n)$ and $u_2 = c^{-1}(n+1)$ we have $d(u_1, v) = c(u_1, v)$ and $d(u_1, v) = c(u_2, v)$, therefore both u_1 and u_2 are distance-consistent. □

Theorem 3.2.2. $dc(C_{2n+1}) = 3$



(a) $dc(C_{2n}) = 2$. The vertices labelled n and $n+1$ are distance-consistent.



(b) $dc(C_{2n+1}) = 3$. The vertices labelled n, $n+1$ and $n+2$ are distance-consistent.

Proof. Let c be a labelling where the vertex labelled 1 is adjacent to the vertex labelled 2, the vertex labelled 2 is adjacent to the verex labelled 3 and so on (see figure 3.1b). $u := c^{-1}(n+1)$ satisfies $d(u,v) = c(u,v)$ for all vertices u, v, and therefore u is distance-consistent. Furthermore the vertices at distance $n+1$ from the vertex labelled n are labelled $2n$ and $2n+1$, and vertices at distance $k < n+1$ from $c^{-1}(n)$ are labelled $n-k$ and $n+k$ respectively. Thus $c^{-1}(n)$ is distance-consistent. Similarly the vertices at distance $n+1$ from the vertex labelled $n+2$ are labelled 1 and 2, and vertices at distance $k < n+1$ from $c^{-1}(n+2)$ are labelled $(n+2-k)$ and $n+2+k$. Thus $c^{-1}(n+2)$ is also distance-consistent, and $dc(C_{2n+1}) \geq 3$.

Assume there is a 4-distance-consistent labelling c of C_{2n+1}. By Proposition 3.1.4 the two vertices which are at distance n from $c^{-1}(1)$ must be distance-consistent and the two vertices at distance n from $c^{-1}(2n+1)$ must be distance-consistent. Furthermore $c^{-1}(1)$ and $c^{-1}(2n+1)$ are not adjacent. Let u_1 and u_2 be the vertices at distance n from $c^{-1}(1)$. The vertex at distance n from u_1 not labelled 1 must be labelled 2 since it cannot be labelled $2n+1$. But so must the vertex at distance n from u_2 not labelled 1. So both neighbours of $c^{-1}(1)$ must be labelled 2 which is impossible, so $dc(C_{2n+1}) \leq 3$. $\square$

3.3 Trees

For trees we shall establish bounds for the local list number and examine some interesting types of trees.

Proposition 3.3.1. *The only trees which are (local) list graphs are paths and conversely all paths are (local) list graphs. If T is a tree then $dc(T) = |V(T)|$ if T is a path, otherwise $dc(T) \leq |V(T)| - 1$.*

Proof. This follows from the characterization of local list graphs in theorem 2.2.3. □

An example of a graph tree with $dc(T) = |V(T)| - 1$ is S_3: the star graph with 3 leaves.

There is a simple lower bound for the local list number of trees.

Proposition 3.3.2. *All non-trivial trees are 2-distance-consistent.*

Proof. Choose a leaf and label it 1 and its neighbor 2. Then go through the vertices at distance 2, then 3 and so on and give each vertex the lowest available label. The resulting labelling is 2-distance-consistent. □

A natural question then is if there is a tree which is not 3-distance-consistent. First we show some useful results.

Proposition 3.3.3. *Let T be a tree and let c be a labelling of T. Assume $u \in V(T)$ is a distance-consistent internal vertex. If $c(u) \notin \{1, |V(T)|\}$ then two neighbours of u must be labelled $c(u) - 1$ and $c(u) + 1$.*

Proof. Since u is an internal vertex it has at least two neighbours. If u has a neighbour v which has $c(v) \notin \{c(u) - 1, c(u) + 1\}$ then either $d(u, c^{-1}(c(u) + 1)) > 1 = d(u, v)$ or $d(u, c^{-1}(c(u) - 1)) > 1 = d(u, v)$ despite $c(u, c^{-1}(u + 1)) = 1 = c(u, c^{-1}(u) - 1) < c(u, v)$. Thus u is not distance-consistent, a contradiction. □

Proposition 3.3.4. *Let c be a vertex labelling of a graph G. Let c' be the labelling defined by $c'(v) = |V(G)| + 1 - c(v)$ for all vertices $v \in V(G)$. Then $u \in V(G)$ is distance-consistent with respect to c' if and only if u is distance-consistent with respect to c.*

Proof. For all vertices v_1, v_2 we have $c(v_1, v_2) = c'(v_1, v_2)$. Therefore u, v, and w satisfy $c(u, v) = c(u, w) + 1$ if and only if $c'(u, v) = c(u, w) + 1$, thus u is distance-consistent with respect to c if and only if u is distance-consistent with respect to c'. □

Now we show that there is a tree which is not 3-distance-consistent.

Theorem 3.3.5. *Start with the star graph S_4. For each leaf v in S_4 add another vertex v' and an edge vv'. The resulting graph T (see figure 3.2) is 2-distance-consistent but not 3-distance-consistent.*

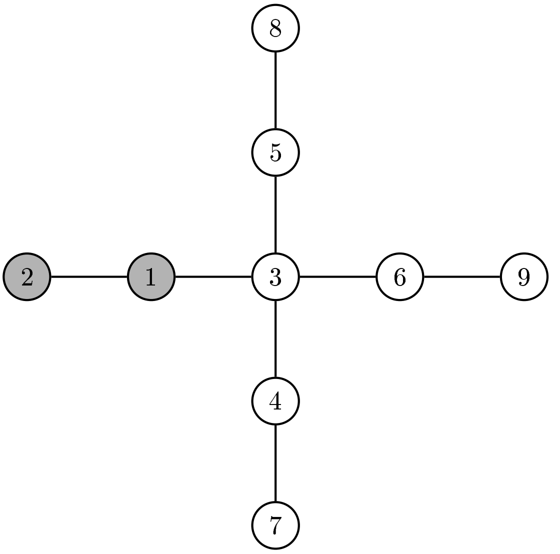


Figure 3.2: Tree with $dc = 2$. The labelling is 2-distance-consistent. Vertices 1 and 2 are distance-consistent.

Proof. Let m be the vertex of degree 4 and let c be a labelling of T.

If $c^{-1}(1)$ is distance-consistent then $c(m) = 1 + d(c^{-1}(1), m) \leq 3$ since m is the only vertex at distance $d(c^{-1}(1), m)$ from $c^{-1}(1)$. Similarly if $c^{-1}(9)$ is distance-consistent, then $c(m) = 9 - d(c^{-1}(9), m) \geq 7$. Thus $c^{-1}(1)$ and $c^{-1}(9)$ are not both distance-consistent.

By Proposition 3.3.7, if a vertex $u_0 \notin c^{-1}(\{1, 9, m\})$ is distance-consistent then $c(m) \in \{c(u_0) - 1, c(u_0) + 1\}$ as u_0 is either a leaf, in which case the neighbour of u_0 must be labelled $c(u_0) \pm 1$ and m must be labelled $c(u_0) \mp 1$ or u_0 is a neighbour of m. Therefore if a vertex $u \in V(G)$ is distance-consistent $c(u) \notin \{1, 9, m\}$ then $c(u) \in \{c(m) - 1, c(m) + 1\}$.

To find a contradiction assume T is 3-distance-consistent. Then there are three distance-consistent vertices u_1, u_1', and u_1''. Without loss of generality assume $u_1 \neq m$ and $c(u_1) \notin \{1, 9\}$ (at most one of the vertices labelled 1 and 9 are distance-consistent).

First we show that if $u_1 := c^{-1}(1)$ or $u_9 := c^{-1}(9)$ are distance-consistent then T is not 3-distance-consistent. Assume u_1 is distance-consistent. Suppose u_1 is internal. One neighbour of u_1 must be labelled 2, call it u_2. $u_1 u_2$ is a cut-edge separating T in two components T_1 and T_2. Without loss of generality assume $u_1 \in T_1$ and $u_2 \in T_2$. No vertex $v \in T_1$ is distance-consistent because

$d(v, u_2) > d(v, u_1)$ but $c(v, u_2) < c(v, u_1)$. $u_3 := c^{-1}(3) \in T_1$ because u_3 must be a neighbour of u_1 and $u_3 \neq u_2$. No vertex $v' \in T_2 \setminus \{u_2\}$ is distance-consistent because $c(v') \geq 4$ and thus $c(v', u_3) < c(v', u_2)$ but $d(v', u_2) < d(v', u_2)$.

Suppose u_1 is instead a leaf. Then the neighbour of u_1 has label 2, m has label $c(m) = 3$, the vertices at distance 3 from u_1 has labels $4, 5, 6$ and the other vertices have labels $7, 8$, and 9. Any distance-consistent vertex $u \neq u_1$ must be labelled $2, 3$, or 4. To show that they are not distance-consistent we must find $v, w \in V(G)$ such that $c(u, v) = c(u, w) + 1$ and $d(u, v) > d(u, w)$. For $c(u) = 3$ we have $c(v) = 1$ and $c(w) = 5$, and for $c(u) = 4$ we have $c(v) = 1$ and $c(w) = 6$. Therefore c is not 3-distance-consistent - a contradiction. We conclude that u_1 is not distance-consistent.

Let c_1 be a labelling where u_1 is distance-consistent. For any such labelling, we can define a labelling c_9 by setting $c_9(v) = |V(G)| + 1 - c_1(v)$ for all vertices v. By Proposition 3.3.4 if u_9 is distance-consistent then c_9 is not 3-distance-consistent since there is a one-to-one correspondence between such labellings, it follows that no labelling is 3-distance-consistent and has a vertex labelled 9 that is distance-consistent.

The only remaining possibility is that the distance-consistent vertices u_1, u_1', and u_1' are not labelled 1 or 9. By the argument in the beginning of the proof, they must be labelled $c(m)-1$, $c(m)$, and $c(m)+1$. Let v_1 be the leaf adjacent to u_1. Since u_1 and m are distance-consistent we have $c(u_1, v_1) = 1$, $c(m, u_1) = 1$, therefore $c(m, v_1) \leq 2$. This is a contradiction as $c(m, v_1) \geq 3$ since m is distance-consistent and has degree 4. □

In appendix A we calculate $dc(T)$ for all trees T on 9 or fewer vertices. The tree in fig 3.2 is the only (and therefore smallest) such tree with local list number 2.

Lemma 3.3.6. *Let G be a graph and $e = xy \in E(G)$ be a cut-edge. If both x and y are distance-consistent then $c(x, y) = 1$.*

Proof. Let X and Y be the vertex sets separated by e, with $x \in X$ and $y \in Y$. If either endpoint has degree 1 we have $c(x, y) = 1$. If both endpoints have degree greater than 1 there exist vertices $x' \in X \setminus \{x\}$ and $y' \in Y \setminus \{y\}$. By contraposition of proposition 3.1.1 :

$$\begin{aligned} d(x, y') > d(x, y) &\implies c(x, y') \geq c(x, y) \\ d(y, x') > d(y, x) &\implies c(y, x') \geq c(y, x). \end{aligned}$$

Thus, for all vertices $v \notin \{x, y\}$ we have $c(x, y) \leq c(x, v)$ and therefore $c(x, y) = 1$. □

Proposition 3.3.7. *Let T be a tree and let $u \in V(T)$ be distance-consistent. u has at most 2 distance-consistent neighbours.*

Lemma 3.3.8. *Let G be a graph. A vertex $v \in G$ is adjacent to at most two distance-consistent leaves. Furthermore the distance-consistent leaves adjacent to v must be labelled $c(v) - 1$ and/or $c(v) + 1$.*

Proof. Let l be a distance-consistent leaf. The neighbour v of l must be labelled $l - 1$ or $l + 1$, otherwise we would have $d(l, c^{-1}(l + 1)) > 1 = d(l, v)$ or $d(l, c^{-1}(l - 1)) > 1 = d(l, v)$ despite $1 = |c(l) - (l \pm 1)| < c(l, v)$ which would contradict 3.1.1. Therefore any distance-consistent leaf adjacent to v must be labelled $c(v) - 1$ or $c(v) + 1$. □

As the list number is in a sense a measure of how close graph is to a local list graph we might expect that trees which are in some sense close to a path to have high list numbers and those far from a path (high branching factor), such as stars, to have low list numbers. In the case of stars we find that this is indeed the case.

Theorem 3.3.9. *Let S_k be the star graph with $k + 1$ vertices. $dc(S_k) = 3$ for all $k \geq 3$. Moreover if the internal vertex is labelled 1 or $k + 1$ the labelling is 2-consistent but not 3-distance-consistent. Otherwise the labelling is 3-distance-consistent.*

Proof. Let c be a labelling of $G = S_k$, let v_0 be the internal vertex, and let $l = c(v_0)$.

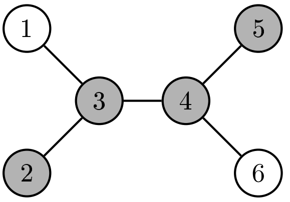
By lemma 3.1.3 v_0 is distance-consistent with respect to any labelling. By lemma 3.3.8, if a vertex $u \in V(G)$ is distance-consistent then $c(u) \in \{l - 1, l, l + 1\}$. Thus $dc(S_k) \leq 3$.

If there is a vertex u_{l-1} with $c(u_{l-1}) = l - 1$ then it is distance-consistent (by contraposition of proposition 3.1.1) since for all vertices $v' \notin \{u_{l-1}, v_0\}$ $d(u_{l-1}, v') = 2$ and $c(u_{l+1}, v') \geq c(u, v_0) = 1$. Similarly if there is a vertex u_{l+1} with $c(u_{l+1}) = l + 1$ then it is also distance-consistent since for all vertices $v'' \notin \{u_{l+1}, v_0\}$ $d(u_{l+1}, v'') = 2$ and $c(u_{l+1}, v'') \geq c(u, v_0) = 1$. If $l \notin \{1, k + 1\}$ both u_{l-1} and u_{l+1} exist and c is 3-distance-consistent. Otherwise precisely one of u_{l-1} and u_{l+1} exist and c is 2-distance-consistent. □

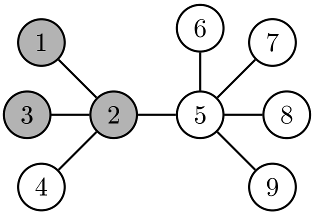
The **double star** $S_{k,l}$ with k and l leaves is the union of two stars S_k and S_l plus an edge between the middle vertices. Take for example $S_{2,2}$ (see figure 3.3a) and $S_{2,3}$ (see figure 3.3b). Computer calculations show that $dc(S_{2,2}) = 4$ and $dc(S_{3,4}) = 3$ (see Appendix A, figure A.1i, and figure A.1ce) We shall see that $S_{2,2}$ is an exception and that the list number of all but a few double stars is 3, just like star graphs. First we show a lower bound for $dc(S_{k,l})$.

Lemma 3.3.10. $dc(S_{k,l}) \geq 3$.

Proof. $S_{1,1}$ is isomorphic to P_4, so $dc(S_{1,1}) = 4$. Assume $\min(k, l) \geq 2$.



(a) $dc(S_{2,2}) = 4$. The vertices labelled $2, 3, 4$, and 5 are distance-consistent.



(b) $dc(S_{3,4}) = 3$. The vertices labelled $1, 2$, and 3 are distance-consistent.

At least one of the middle vertices has at least two neighbours. Call such a vertex u_2, label it 2 and label the adjacent leaves 1 and 3 through $d_G(u_2)$. Label the other middle vertex $d_G(u_2) + 1$. Finally label the remaining vertices arbitrarily and denote the obtained labelling by c.

u_2 is distance-consistent as its neighbours are labelled 1 and 3 through $d_G(u_2)+1$ and the vertices at distance 2 from u_2 are labelled $d_G(u_2+1)$ through $|V(G)|$.

$c^{-1}(1)$ is trivially distance-consistent (c follows the labelling scheme used to prove Proposition 3.1.2).

The vertex $u_3 := c^{-1}(3)$ is also distance-consistent as its neighbour is labelled 2, the vertices at distance 2 from u_3 are labelled 1 and 4 through $d_G(u_2)+1$, and vertices at distance 3 from u_3 are labelled $d_G(u_2)+2$ through $|V(G)|$. Therefore, $S_{k,l}$ is 3-distance-consistent. $\square$

Lemma 3.3.11. *Let c be a labelling of G and let $v \in V(G)$. If two leaves u_1 and u_2 adjacent to v are distance-consistent, then no vertex distinct from u_1, u_2 and v is distance-consistent.*

Proof. To find a contradiction assume there is a distance-consistent vertex $u' \notin \{u_1, u_2, v\}$. By lemma 3.3.8 u_1 and u_2 must be labelled $c(v) + 1$ and $c(v) - 1$. But this gives a contradiction since either $c(u', v) < c(u', u_1)$ or $c(u', v) < c(u', u_2)$, but $d(u', v) < d(u', u_1) = d(u', u_2) = d(u', v) + 1$. $\square$

Theorem 3.3.12. *$dc(S_{k,l}) = 3$ if and only if $\max\{k, l\} \geq 3$.*

Proof. Let $G \simeq S_{k,l}$, with $k, l \geq 3$. We already know that $dc(G) \geq 3$. To find a contradiction assume there is a 4-distance-consistent labelling c. By lemma 3.3.11 only two leaves are distance-consistent as there are only two internal vertices. Thus both internal vertices are distance-consistent and each of the middle vertices is adjacent to precisely one distance-consistent leaf, otherwise one internal vertex would have three distance-consistent neighbours contradicting Proposition 3.3.7. By lemma 3.3.6 there must be an n such that the distance-consistent vertices are labelled n, $n+1$, $n+2$, and $n+3$. For each $k \in \{n, \dots n+3\}$ let $u_k := c^{-1}(k)$. We have $P = u_n u_{n+1} u_{n+2} u_{n+3} \subseteq G$.

Pick u_n to be the root of the tree. Since u_{n+3} is at depth 3, the vertices at depth 2 must have labels m satisfying $|n-m| \leq 3$, otherwise we would have a contradiction to our assumption. So since $k, l \geq 3$, u_{n+1} must have neighbours labelled $n-1$ and $n-2$, but then u_{n+1} cannot be distance-consistent because $|c(u_{n+1}) - (n-2)| = 3 = c(u_{n+1}, u_{n+3}) + 1$ while $d(u_{n+1}, u_{n+3}) = 2 > 1 = d(u_{n+1}, c^{-1}(n-2))$. □

Computer calculations show that $dc(S_{1,2}) = dc(S_{2,2}) = 4$, see appendix A.

Theorem 3.3.13. *For each integer $n \geq 3$ and $3 \leq k \leq n$ there is a tree on n vertices which has local list number k.*

Proof. Some different constructions are needed depending on how large k is compared to n. For $k = 3$ S_n is an example of a tree on n vertices with local list number k. Assume $k \geq 4$.

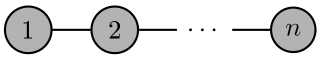
Suppose $n = k+1$. A k-distance-consistent tree is constructed as follows: Start with the path $P_k = v_1 v_2 \dots v_k$ and add a vertex v_{k+1} and the edge $v_{k-1}v_{k+1}$. The resulting graph has a k-distance-consistent labelling $c(v_j) = j$, but is not $(k+1)$-distance-consistent as it is not a path.

Suppose $k+2 \leq n \leq 2k-2$. A k-distance consistent graph $T_{n,k}$ can be constructed as follows. Start with the paths $P = v_1 v_2 \dots v_k$ and $P' = v_{k+1} v_{k+2} \dots v_n$ and add the edge $v_{k-1}v_{k+1}$. See Figure 3.4c. The labelling $c(v_i) := i$ is k-distance-consistent with v_1 through v_k being the only distance-consistent vertices as

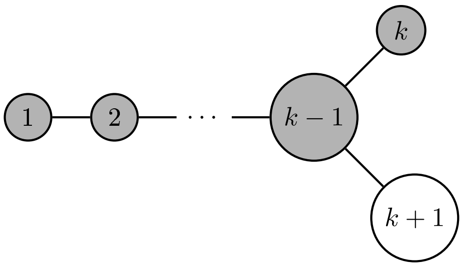
$$d(v_i, v_j) = \begin{cases} c(v_i, v_j) & \text{if } i = k \text{ or } j = k \\ c(v_i, v_j) & \text{if } i \leq k-1 \text{ and } j \leq k-1 \\ c(v_i, v_j) & \text{if } i \geq k+1 \text{ and } j \geq k+1 \\ c(v_i, v_j) - 1 & \text{if } i \leq k-1 \text{ and } j \geq k+1 \\ c(v_i, v_j) - 1 & \text{if } j \leq k-1 \text{ and } i \geq k+1 \end{cases}.$$

Assume there is a $(k+1)$-distance-consistent labelling c' of $T_{n,k}$. Then there are distance-consistent vertices $v_a \in P - v_k$ and $v_b \in P'$ since these paths are shorter than or equal to k. As v_n is the only vertex at maximum distance from v_a it must be labelled 1 or n (see Proposition 3.1.4). Using proposition 3.3.4 we assume $c'(v_n) = n$ without loss of generality. Similarly, v_1 is the only vertex at maximum distance from v_b, so it must be labelled 1.

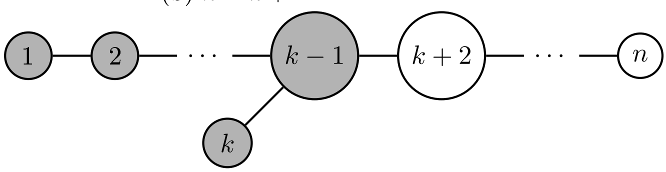
At least one of $P \setminus \{v_k\}$ and $P' \cup v_{k-1}$ must have at least $k/2$ distance-consistent vertices. Suppose $P \setminus \{v_k\}$ has $k/2$ distance-consistent vertices. Then there exists a distance-consistent vertex $v_c \in P \setminus \{v_k\}$ with $d(v_c, v_1) \geq$



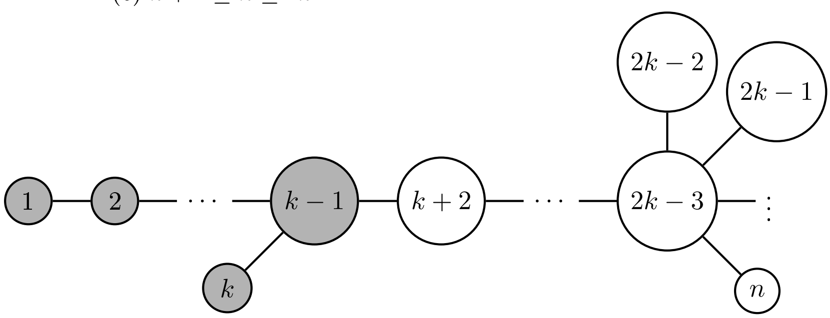
(a) $n = k$



(b) $n = k + 1$



(c) $k + 2 \leq n \leq 2k - 2$



(d) $2k - 2 < n < 3k$

Figure 3.4: Trees on n vertices with $dc(G) = k$. The labellings depicted are k-distance-consistent for the indicated values of n.

$d(v_c, v_{k-1})$. Take $v_j \in P \setminus \{v_k\}$. Since P' is a path, v_c is distance-consistent and $d(v_c, v_j) \leq d(v_c, v_1)$, $c'(v_j, v_c) = d(v_j, v_c) = |j - c|$.By contraposition of Proposition 3.1.1 if $d(v_b, v_j) < d(v_b, v_c)$ then $c'(v_b, v_j) \leq c'(v_b, v_c)$. Otherwise, if $d(v_b, v_j) > d(v_b, v_c)$ then $c'(v_b, v_j) \geq c'(v_b, v_c)$ therefore $c'(v_j) = j$.

There are two possibilities for $c'(v_k)$. Either $c'(v_k) = k$ and $c' = c$, which is a contradiction as c is not $(k+1)$-distance-consistent, or $c'(v_k) = k + 1$. The second case also leads to a contradiction as v_b is distance-consistent and $d(v_b, v_k) > d(v_b, v_{k-1})$ but $c'(v_b, v_k) = c'(b) - (k+2) \not\geq c'(b) - (k-1) = c'(v_b, v_{k-1})$.

If instead $P' \cup \{v_{k-1}\}$ has $k/2$ distance consistent vertices we reach a contradiction using a completely analogous method.

Suppose instead $n > 2k - 2$. Let $T_{n,k}$ be the tree constructed as follows. Begin with $T_{2k-2,k}$, add $n-(2k-2)$ vertices $v_{2k-1}, v_{2k}, \ldots v_n$ and for all $2k-1 \leq j \leq n$ add the edge $v_{2k-3}v_j$. Let $c(v_i) := i$. See Figure 3.4d. As in the previous case the vertices labelled 1 through k are distance-consistent.

Assume there exists a $(k+1)$-distance-consistent labelling c' of $T_{n,k}$. By lemma 3.3.11 at most one of the leaves $\{v_{2k-2}, \ldots, v_n\}$ is distance-consistent as $k+1 > 3$. If one of these leaves is distance-consistent we can assume it is in P' without loss of generality.

Now we can use the same strategy as in the previous case. There exists distance-consistent vertices $v_a \in P - v_k$ and $v_b \in P'$ since $|V(P - v_k)| = |V(P')| = k - 1$. As v_1 is the only vertex at maximum distance from v_b it must be labelled 1 or n (see Proposition 3.1.4). Using proposition 3.3.4 we assume $c'(v_1) = 1$ without loss of generality. At least one of $P \setminus \{v_k\}$ and $P' \cup v_{k-1}$ must have at least $k/2$ distance-consistent vertices.

First, suppose $P \setminus \{v_k\}$ has $k/2$ distance-consistent vertices. Then there exists a distance-consistent vertex $v_c \in P \setminus \{v_k\}$ with $d(v_c, v_1) \geq d(v_c, v_{k-1})$. Take $v_j \in P \setminus \{v_k\}$. Since P' is a path, v_c is distance-consistent and $d(v_c, v_j) \leq d(v_c, v_1)$. $c'(v_j, v_c) = d(v_j, v_c) = |j - c|$.By contraposition of Proposition 3.1.1 if $d(v_b, v_j) < d(v_b, v_c)$ then $c'(v_b, v_j) \leq c'(v_b, v_c)$. Otherwise, if $d(v_b, v_j) > d(v_b, v_c)$ then $c'(v_b, v_j) \geq c'(v_b, v_c)$. Therefore $c'(v_j) = j$.

As in the previous case there are two possibilities for $c'(v_k)$. Either $c'(v_k) = k$ and $c' = c$, or $c'(v_k) = k + 1$. Both lead to contradictions in the same was as when $n \leq 2k - 2$.

If instead $P' \cup \{v_{k-1}\}$ has $k/2$ distance consistent vertices we reach a contradiction using a similar method. We find that there exists a distance-consistent vertex v'_c with $d(v'_c, v_n) \geq d(v'_c, v_{k-1})$ and conclude that $c'(v_j) = j$ for all $j \geq k + 1$ and $c'(v_{k-1}) = k$. Then $c'(v_k) = k - 1$ or $c'(v_k) = k - 2$. In both cases we reach a contradiction in the same way as when $n \leq 2k - 2$. $\square$

3.4 Bipartite Graphs

Theorem 3.4.1. *(Due to Casselgren) [4] If $n \geq 4$ and $m \geq 4$ then $dc(K_{n,m}) = 1$.*

Proof. Based on a proof sketch provided by Casselgren [4].

Let c be a labelling of $K_{n,m}$, and let X and Y be the partite sets of $G = K_{n,m}$. If $c^{-1}(1) \in X$ is distance-consistent then the vertices in Y must be labelled 2 through $m+1$, and by Proposition 3.1.4 no vertex in Y can be distance-consistent. Also, no other vertex $x \in X$ $(x \neq c^{-1}(1))$ can be distance consistent because at least one vertex $x' \in X$ has $c(x, x') = 1$, but there must be a vertex $y \in Y$ $c(x, y) > 1$. Similarly if $c^{-1}(n+m) \in X$ is distance consistent then the vertices in Y must be labelled $|X|$ through $m+n-1$, and no vertex in Y is distance-consistent, by prop 3.1.4. No vertex $x'' \in X$ other than $c^{-1}(n+m)$ is distance-consistent as there exists at vertex $x''' \in X$ with $c(x'', x''') = 1$.

For sake of contradiction assume c is 2-distance-consistent. Then there are two distance-consistent vertices u and u' in $V(G)$. Without loss of generality assume $u \in X$ and that $c(u') > c(u)$. Since u and u' are both distance-consistent we have $c(u) \notin \{1, a+b\}$ and $c(u') \notin \{1, a+b\}$.

Suppose first that $u' \in X$. Since $n, m \geq 4$ there exists a vertex $v \in Y$ with $c(v) < c(u) < c(u')$. This is a contradiction since $c(u', v) > c(u', u)$, but $d(u', v) = 1 < d(u, v) = 2$.

Suppose instead that $u' \in Y$. If $c(u') = c(u)+1$ then $w := c^{-1}(c(u)+2) \in Y$ since $u \in X$ and $|Y| \geq 4$. This leads to a contradiction because we must also have $w \in X$ since $c(u', w) = 1$ and u' has at least 4 neighbours. If instead $c(u') > c(u) + 1$ then $w_2 := c^{-1}(c(u)+1) \in Y$, which is also a contradiction as $c(u', w) < c(u', u)$ but $d(u', w) = 2 > d(u', u) = 1$. We must conclude that c is not 2-distance-consistent and thus $dc(K_{n,m}) = 1$. □

These large complete bipartite graphs are an excellent example showing that there are infinite families of graphs that are not 2-distance-consistent.

This leaves only $K_{1,n}$, $K_{2,n}$ and $K_{3,n}$. $K_{1,n}$ is isomorphic to the star graph S_n and so $dc(K_{1,n}) = 3$ for all $n \geq 2$.

Theorem 3.4.2. *For all $k \geq 2$ $dc(K_{2,k}) = 2$*

Proof. Let (X, Y) be the partition of $G := K_{2,k}$ where $|X| = 2$ and $|Y| = k$ Label the vertices in the partite set of size two 1 and $k+2$, and label the remaining vertices arbitrarily. It is easy to check that the vertices labelled 1 and 2 are distance-consistent.

If $u_0 \in Y$ is distance-consistent and not labelled 1 or $k+2$ then the vertices in X must be labelled $c(u_0)+1$ and $c(u_0)-1$. So no other vertex in Y, other than those labelled 1 or $k+2$ can be distance-consistent.

If $c^{-1}(1) \in Y$ is distance-consistent the vertices in X must be labelled 2 and 3, and the remaining vertices must be labelled 4 through $k+2$. Thus, no other vertex in Y can be distance-consistent. The vertices labelled 2 and 3 cannot be distance-consistent either, since $u := c^{-1}(2)$, $v := c^{-1}(4)$, $w := c^{-1}(3)$ as well as $u' := c^{-1}(3)$, $v' := c^{-1}(1)$, and $w' := c^{-1}(2)$ contradict the local list criterion.

Similarly, if $c^{-1}(k+2) \in Y$ is distance-consistent the vertices in X must be labelled $k-1$ and k, and the remaining must be labelled 1 through $k-2$. Thus, no other vertex in Y can be distance-consistent. The vertices in X cannot be distance-consistent either, since $c(u) = k+1$, $c(v) = k-1$, $c(w) = k$ as well as $c(u) = k$, $c(v) = k-1$, and $c(w) = k+1$ contradict the local list criterion.

So if c is a labelling with a distance-consistent vertex in Y it is not 2-distance-consistent, and therefore $dc(K_{2,k}) \leq 2$. □

Proposition 3.4.3. $dc(K_{3,3}) = 2$

Proof. Let X and Y be the partite sets of $G = K_{3,3}$. Let c be the labelling of G where the vertices in X are labelled $1, 2$, and 4, and the vertices in Y are labelled $3, 5$, and 6.

The vertex labelled 3 is distance consistent because its neighbours are at label distance 2 or less from $c^{-1}(3)$ while the vertices at distance 2 from $c^{-1}(3)$ are at label distance at least 3. Similarly the vertex labelled 4 is distance-consistent because all $v \in Y$ satisfy $c(c^{-1}(4), v) \leq 2$ while all $v' \in X$ satisfy $c(c^{-1}(4), v') \geq 2$.

Suppose there is a 3-distance-consistent labelling c' of G. If there a partite set X' with 3 distance-consistent vertices then by Proposition 3.1.4 there must be a vertex in X' set labelled 1 or 6. But because this vertex is also distance consistent another vertex in X' must also be labelled 1 or 6. Since $c'^{-1}(1)$ is distance consistent the last vertex $v \in X'$ must be labelled 5, but since $c'^{-1}(6)$ is distance-consistent v must also be labelled 1, which is a contradiction.

This leaves only the option that one partite set set X_1 has at least one distance-consistent vertex and the other partite set X_2 has two distance-consistent vertices. By Proposition 3.1.4 the vertices labelled 1 and 6 must be in different partite sets as both partite set contain distance-consistent vertices. Neither $c'^{-1}(1)$ nor $c'^{-1}(6)$ can be distance-consistent because then they would need to be in the same partite set.

If $c^{-1}(2) \in X$ is distance-consistent the vertices in X must be labelled $2, 5$, and 6. But then, by Proposition 3.1.4 neither $c^{-1}(5)$ nor $c^{-1}(3)$ can be distance-consistent as they are not in the same partite set as the vertex at the furthers label-distance from them. There are four vertices which are not distance-consistent, which contradicts c being 3-distances-consistent, therefore $c^{-1}(2)$ is not distance consistent.

Finally, assume $c^{-1}(5)$ is distance-consistent. Let Y be the partite set such that $c^{-1}(5) \in Y$. The vertices in Y must be labelled $1, 2$ and 5. By 3.1.4 $c^{-1}(4)$ cannot be distance consistent as it is not in the same partite set as $c^{-1}(1)$. Again we find that the vertices labelled $1, 2, 4$ and 6 are not distance-consistent contradicting c being 3-distance-consistent. Therefore G is at most 2-distance-consistent.

Theorem 3.4.4. *For all $k \geq 4$ $dc(K_{3,k}) = 1$.*

Proof. Let (X, Y) be the partition of $K_{3,k}$ where $|X| = 3$ and $|Y| = k$. Assume one vertex u is distance-consistent. We shall show that no other vertex is distance-consistent.

Suppose first $u \in Y$. If $c(u) = 1$ then $c(X) = \{2, 3, 4\}$. If $c(u) = k + 3$ then $c(X) = \{k, k+1, k+2\}$. If $c(u) \notin \{1, k+3\}$ then either $c(X) = \{c(u) - 1, c(u) + 1, c(u) + 2\}$ or $c(X) = \{c(u) - 1, c(u) - 2, c(u) + 1\}$. For any of these possible sets $c(X)$ there is only one possible value for $c(u)$, and thus no other vertex in Y can be distance-consistent. In either case each vertex $x \in X$ is adjacent to the vertex at maximum label distance from x, therefore, by Proposition 3.1.4 x is not distance-consistent.

Suppose instead $u \in X$. We have already shown that if a vertex in X is distance-consistent then it is the only distance-consistent vertex, therefore the vertices in Y are not distance-consistent.

If $c(u) = 1$ then the vertices in Y must be labelled 2 through $k + 1$ and the other vertices in X must be labelled $k + 2$ and $k + 3$. Then $c^{-1}(k + 2)$ is not distance consistent as $c(c^{-1}(k+2), c^{-1}(k) = 2 = c(c^{-1}(k+2), c^{-1}(k+3)+1$ but $d(c^{-1}(k + 2), c^{-1}(k + 3)) = 2 > d(c^{-1}(k + 2), c^{-1}(k)) = 1$. Similarly $c^{-1}(k + 3)$ is not distance consistent as $c(c^{-1}(k+3), c^{-1}(k+1) = 2 = c(c^{-1}(k+3), c^{-1}(k+2) + 1$ but $d(c^{-1}(k + 3), c^{-1}(k + 2)) = 2 > d(c^{-1}(k + 3), c^{-1}(k + 1)) = 1$.

If instead $c(u) = k + 3$ then the vertices in Y must be labelled 3 through $k + 2$ and the other vertices in X must be labelled 1 and 2. Then $c^{-1}(2)$ is not distance consistent as $c(c^{-1}(2), c^{-1}(4) = 2 = c(c^{-1}(2), c^{-1}(1) + 1$ but $d(c^{-1}(2), c^{-1}(1)) = 2 > d(c^{-1}(2), c^{-1}(4)) = 1$.

Finally, if $c(u) \notin \{1, k + 3\}$ then at least one of $c^{-1}(1)$ and $c^{-1}(k + 3)$ in in X. The final vertex v in X must be labelled $1, 2, k+2$, or $k+3$ since it is one of only two vertices at maximum distance from u. If $c(v) = 1$ or $c(v) = k + 3$ then v is not distance consistent as u is distance-consistent and $u \neq v$. If $c(v) = 2$ then $c(X) = \{1, 2, c(u)\}$ and v is not distance-consistent as $c(v, c^{-1}(4)) = 2 = c(v, c^{-1}(1)) + 1$ but $d(v, c^{-1}(1)) = 2 > d(v, c^{-1}(4)) = 1$. If $c(v) = k + 2$ then $c(X) = \{c(u), k + 2, k + 3\}$ and v is not distance consistent as $c(v, c^{-1}(k) = 2 = c(v, c^{-1}(k + 3) + 1$ but $d(v, c^{-1}(k + 3)) = 2 > d(c^{-1}(k + 2), c^{-1}(k)) = 1$.

In conclusion for each choice of $c(u)$ u is the only distance consistent vertex. $\square$

We now know the local list number of all complete bipartite graphs. To summarise:

$$dc(K_{n,m}) = \begin{cases} 3 & \text{if } \min\{n,m\} = 1 \text{ and } n \neq m \\ 2 & \text{if } \min\{n,m\} = 2 \text{ or } n = m = 3 \text{ or } n = m = 1 \\ 1 & \text{Otherwise} \end{cases} .$$

Bibliography

[1] M.A. Breuer. Coding the vertexes of a graph. *IEEE Trans. of Info. Theory*, 12(2):148–153, 1966.

[2] M.A. Breuer and J. Folkman. An unexpected result on coding vertices of a graph. *J. Math. Anal. Appl.*, 20(3):583–600, 1967.

[3] S. Pérennes C. Gavoille, D.Peleg and R. Raz. Distance labeling in graphs. *Journal of algorithms*, 54:85–112, 2001.

[4] C.J. Casselgren. Remarks on distance consistent vertex labelings. Private Notes.

[5] R. Diestel. *Graph Theory*. Springer, 5 edition, 2017.

[6] Joseph A Gallian. A dynamic survey of graph labeling. *Electronic Journal of combinatorics*, 1(DynamicSurveys):DS6, 2022.

[7] H. Lennerstad. Characterizing graph classes by distance-consistent labelings. pages –, 2020.

[8] H. Lennerstad and M. Eriksson. List graphs and distance-consistent node labelings. *EJGTA*, 6(1):152–165, 2018.

[9] M.Thorup S. Alstrup, H. Kaplan and U.Zwick. Adjacency labeling schemes and induced universal graphs. *SIAM J. DISCRETE MATH*, 33(1):116–137, 2019.

Appendix A

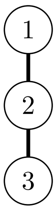
Computer calculations

The following code calculates the local list number of all trees on 9 or fewer vertices using the Python-based free and open source computer algebra system Sage. The results of this code is presented in figure A.1.

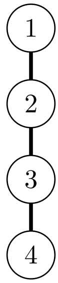
```
import itertools

def vwCombs(u,n):
    # Finds combinations v,w for checking the
    # local list critereon.
    # Runs in O(n)
    combs = set([])
    for v in [x for x in range(1,n+1) if x != u]:
        LHS = abs(u-v)
        w1 = u+LHS-1
        w2 = u-LHS+1
        if w1 > 0 and w1 <= n:
            combs.add((v,w1))
        if w2 > 0 and w2 <= n:
            combs.add((v,w2))
    return combs

def check_dc(G, c):
    # Given graph G and labeling c returns list of
    # distance consistent veritces
    H = G.copy()
    H.relabel(c)
```

```
    dc = []
    for u in H.vertices():
        is_dc = True
        for (v,w) in vwCombs(u, H.num_verts()):
            if H.distance(u,w) > H.distance(u,v):
                is_dc = False
                break
        if is_dc:
            dc.append(u)
    return dc

def makeLabeling(lst):
    # Takes a list of labels and returns a labelling
    # function
    return lambda v: lst[v-1]

# We calculate the local list number of all trees on 9
# or fewer vertices, dc(T), the optimal labelling and
# distance-consistent vertices are added the list trees
trees = []
for n in range(3,10):
    for i,T in enumerate(graphs.trees(n)):
        T.relabel(lambda v:v+1) # Changes initial labels
                                # from 0 through n-1
                                # to 1 through n
        dcMax = 0
        optimalLabel = []
        # We simply go through all possible labelligs
        # and calculate the number of
        # distance-consistent vertices
        for label in itertools.permutations(
            list(range(1,n+1))):
            dc=check_dc(T,makeLabeling(list(label)))
            if len(dc) > dcMax:
                dcMax = len(dc)
                optimalLabel += [[label, dc]]
        optimalLabel = list(
            filter(lambda lst: \
                len(lst[1])==dcMax, optimalLabel))
        trees += [(T,dcMax, optimalLabel)]
```

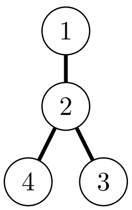
Figure A.1: Trees on n vertices, their local list number and optimal labelling



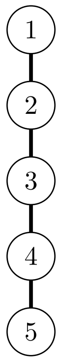
(a) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



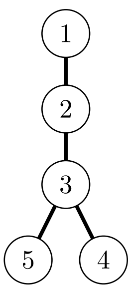
(b) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 2, 3, 4, 1



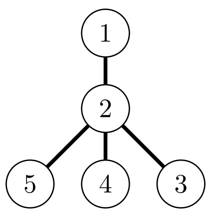
(c) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



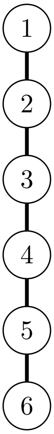
(d) $\mathrm{dc}(T) = 5$. Distance-consistent vertices: 3, 2, 1, 4, 5



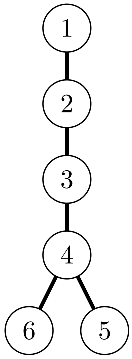
(e) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 4



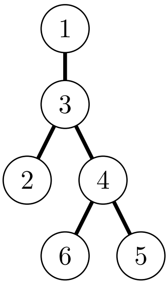
(f) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



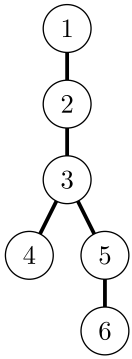
(g) dc(T) = 6. Distance-consistent vertices: 3, 4, 5, 6, 2, 1



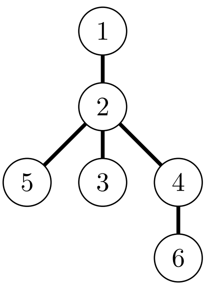
(h) dc(T) = 5. Distance-consistent vertices: 3, 4, 5, 2, 1



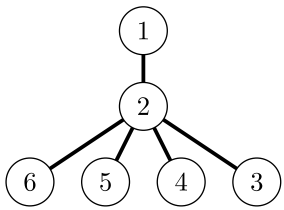
(i) dc(T) = 4. Distance-consistent vertices: 3, 4, 5, 2



(j) dc(T) = 4. Distance-consistent vertices: 3, 2, 1, 4

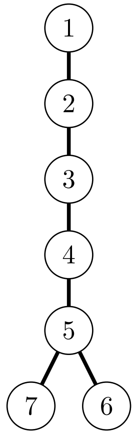


(k) dc(T) = 3. Distance-consistent vertices: 2, 1, 3

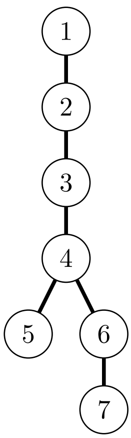


(l) dc(T) = 3. Distance-consistent vertices: 2, 1, 3

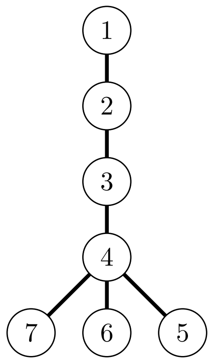
(m) $\mathrm{dc}(T) = 7$. Distance-consistent vertices: 4, 3, 2, 1, 5, 6, 7



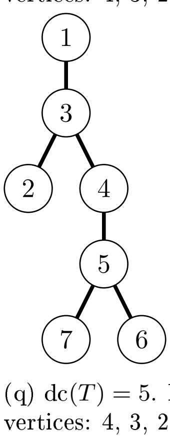
(n) $\mathrm{dc}(T) = 6$. Distance-consistent vertices: 4, 3, 2, 1, 5, 6



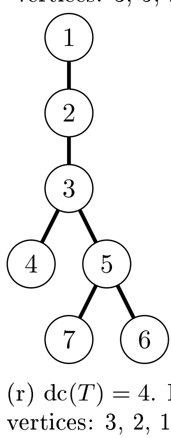
(o) $\mathrm{dc}(T) = 5$. Distance-consistent vertices: 4, 3, 2, 1, 5



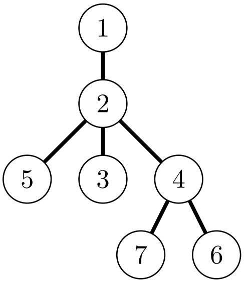
(p) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 5, 2, 1



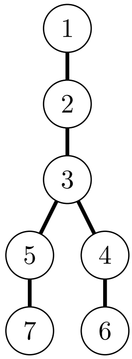
(q) $\mathrm{dc}(T) = 5$. Distance-consistent vertices: 4, 3, 2, 5, 6



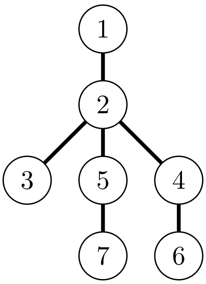
(r) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 4



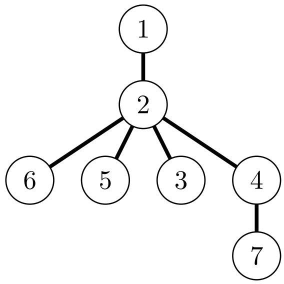
(s) dc(T) = 3. Distance-consistent vertices: 2, 1, 3



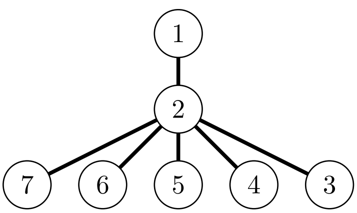
(t) dc(T) = 3. Distance-consistent vertices: 3, 2, 1



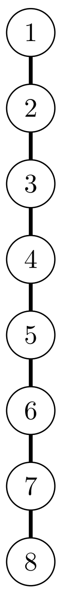
(u) dc(T) = 3. Distance-consistent vertices: 2, 1, 3



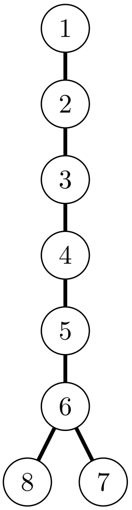
(v) dc(T) = 3. Distance-consistent vertices: 2, 1, 3



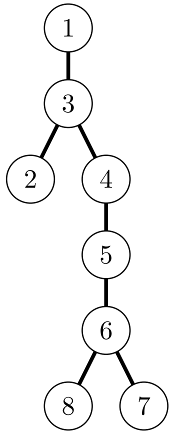
(w) dc(T) = 3. Distance-consistent vertices: 2, 1, 3



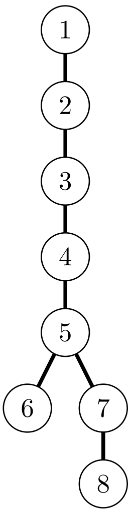
(x) dc(T) = 8. Distance-consistent vertices: 4, 5, 6, 7, 8, 3, 2, 1



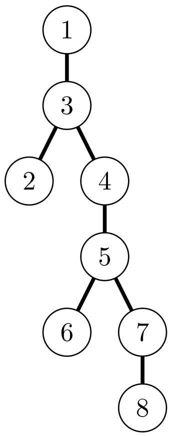
(y) dc(T) = 7. Distance-consistent vertices: 4, 5, 6, 7, 3, 2, 1



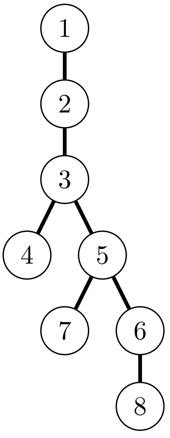
(z) dc(T) = 6. Distance-consistent vertices: 4, 5, 6, 7, 3, 2



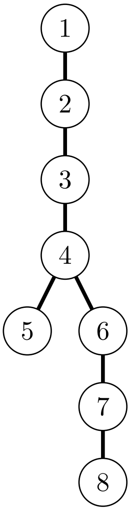
(aa) dc(T) = 6. Distance-consistent vertices: 4, 5, 6, 3, 2, 1



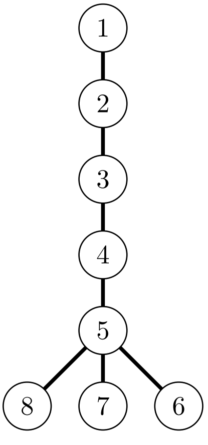
(ab) dc(T) = 5. Distance-consistent vertices: 4, 5, 6, 3, 2



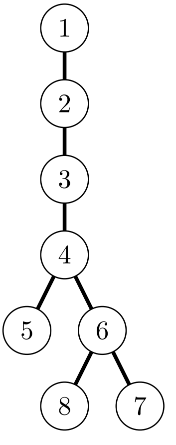
(ac) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 4



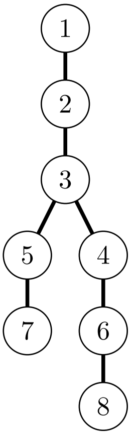
(ad) $\mathrm{dc}(T) = 5$. Distance-consistent vertices: 4, 3, 2, 1, 5



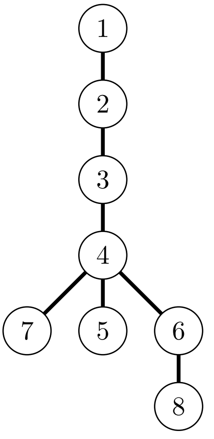
(ae) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 6



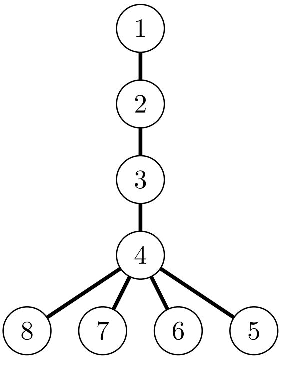
(af) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 5



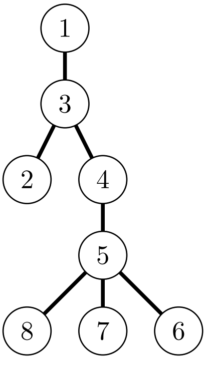
(ag) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 3, 2, 1



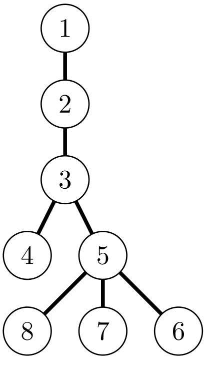
(ah) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 5



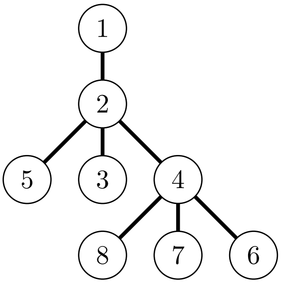
(ai) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 5, 2, 1



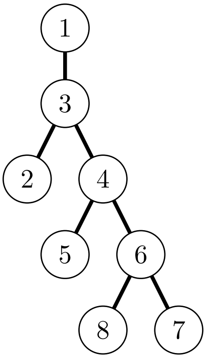
(aj) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 4, 6, 3, 2



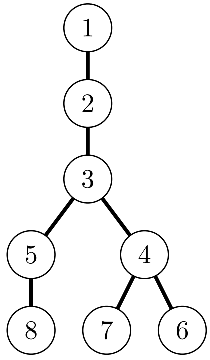
(ak) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 4



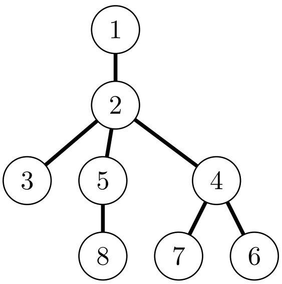
(al) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



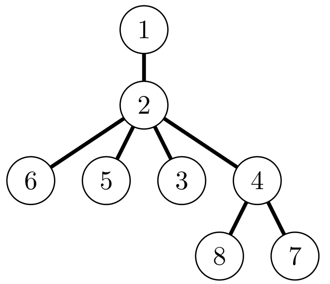
(am) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 4, 3, 2, 5



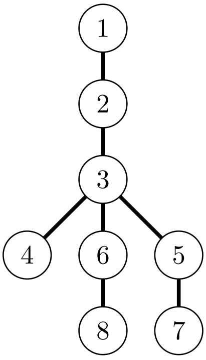
(an) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 3, 2, 1



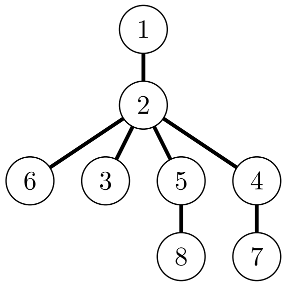
(ao) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



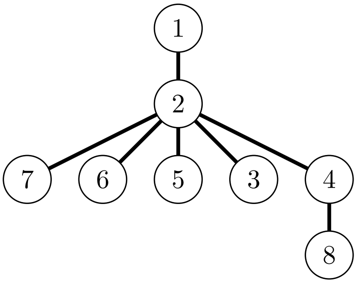
(ap) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



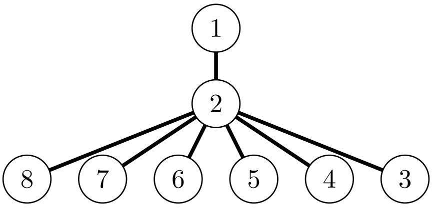
(aq) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 4



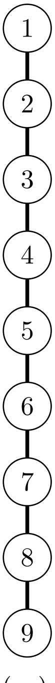
(ar) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



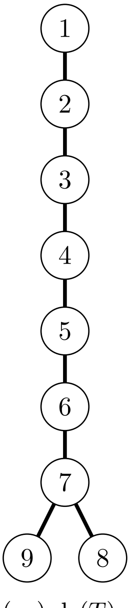
(as) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



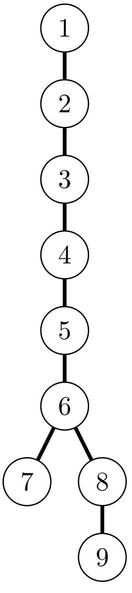
(at) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



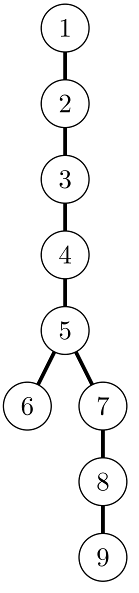
(au) $\mathrm{dc}(T) = 9$. Distance-consistent vertices: 5, 4, 3, 2, 1, 6, 7, 8, 9



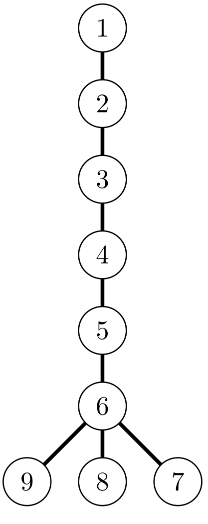
(av) $\mathrm{dc}(T) = 8$. Distance-consistent vertices: 5, 4, 3, 2, 1, 6, 7, 8



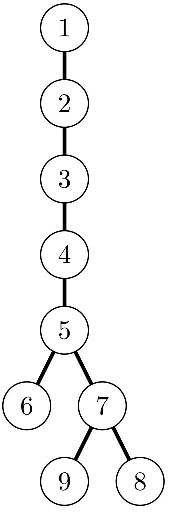
(aw) $\mathrm{dc}(T) = 7$. Distance-consistent vertices: 5, 4, 3, 2, 1, 6, 7



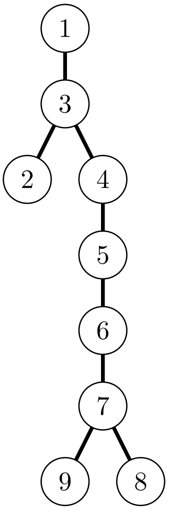
(ax) $\mathrm{dc}(T) = 6$. Distance-consistent vertices: 5, 4, 3, 2, 1, 6



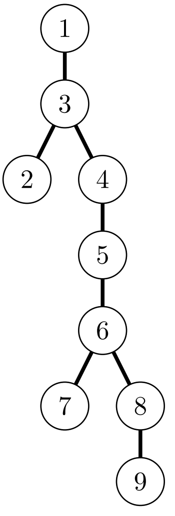
(ay) $\mathrm{dc}(T) = 5$. Distance-consistent vertices: 4, 7, 3, 2, 1



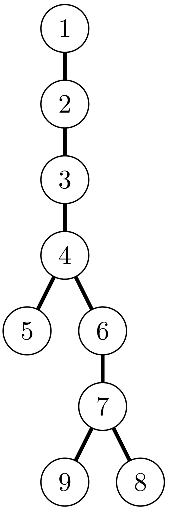
(az) $\mathrm{dc}(T) = 5$. Distance-consistent vertices: 4, 6, 3, 2, 1



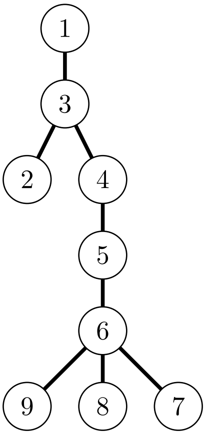
(ba) $\mathrm{dc}(T) = 7$. Distance-consistent vertices: 5, 4, 3, 2, 6, 7, 8



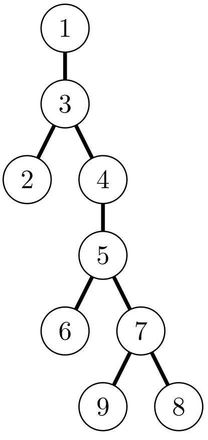
(bb) $\mathrm{dc}(T) = 6$. Distance-consistent vertices: 5, 4, 3, 2, 6, 7



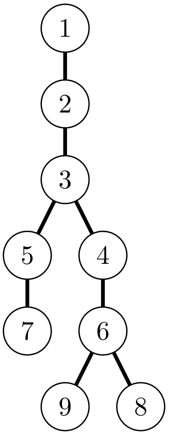
(bc) $\mathrm{dc}(T) = 5$. Distance-consistent vertices: 4, 3, 2, 1, 5



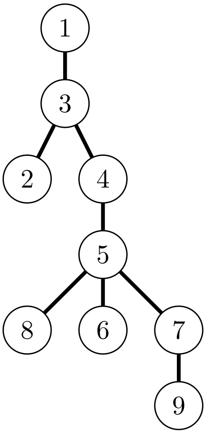
(bd) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 4, 3, 2, 7



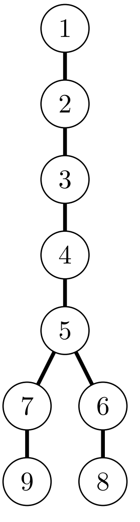
(be) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 4, 3, 2, 6



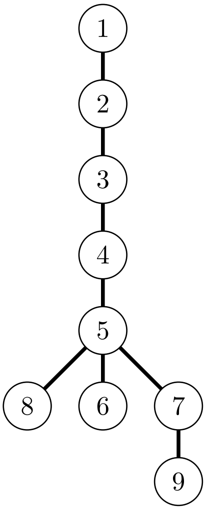
(bf) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 3, 2, 1



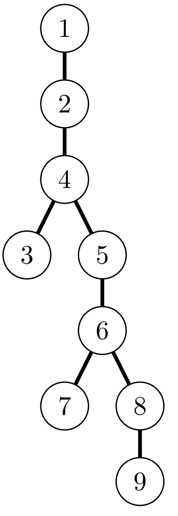
(bg) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 4, 3, 2, 6



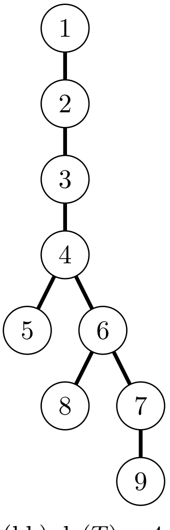
(bh) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 4, 3, 2, 1



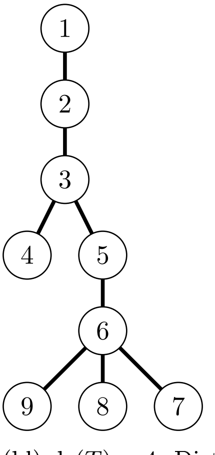
(bi) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 6, 3, 2, 1



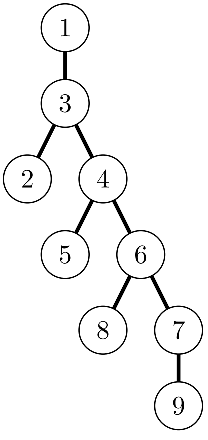
(bj) $\mathrm{dc}(T) = 5$. Distance-consistent vertices: 5, 4, 3, 6, 7



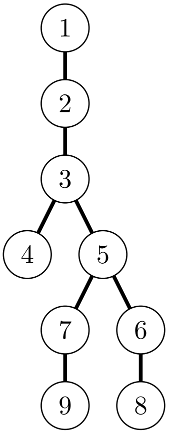
(bk) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 5



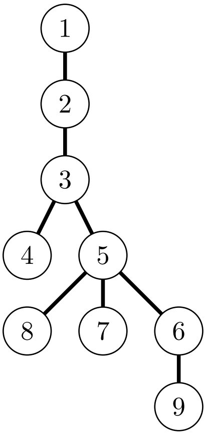
(bl) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 4



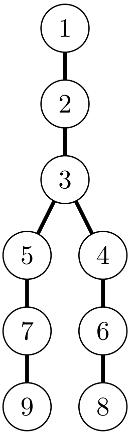
(bm) dc(T) = 4. Distance-consistent vertices: 4, 3, 2, 5



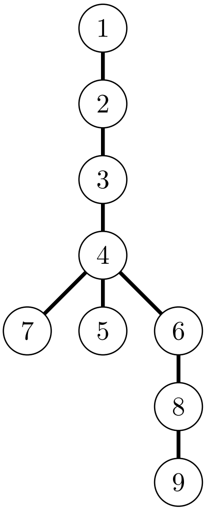
(bn) dc(T) = 4. Distance-consistent vertices: 3, 2, 1, 4



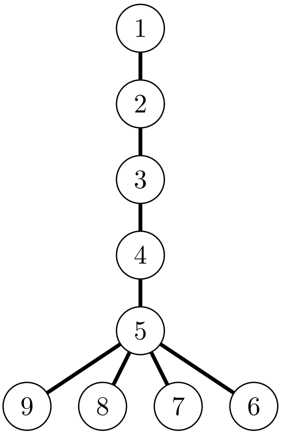
(bo) dc(T) = 4. Distance-consistent vertices: 3, 2, 1, 4



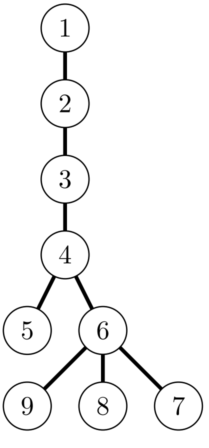
(bp) dc(T) = 3. Distance-consistent vertices: 3, 2, 1



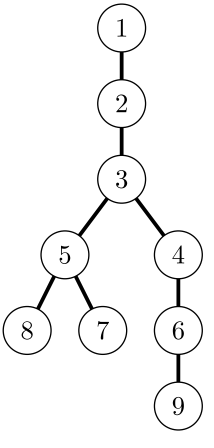
(bq) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 5



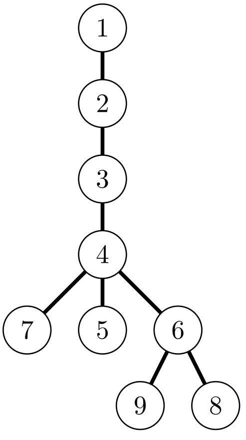
(br) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 6



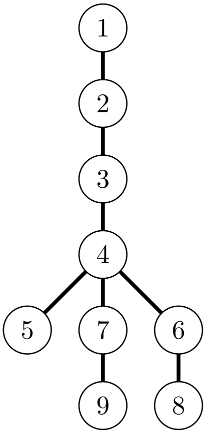
(bs) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 5



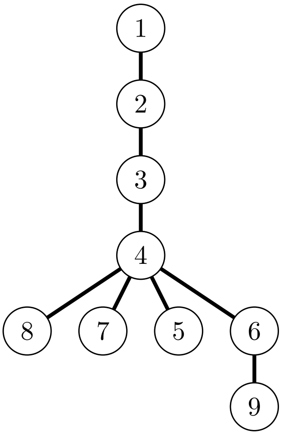
(bt) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 3, 2, 1



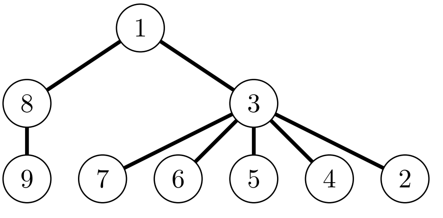
(bu) dc(T) = 4. Distance-consistent vertices: 3, 2, 1, 5



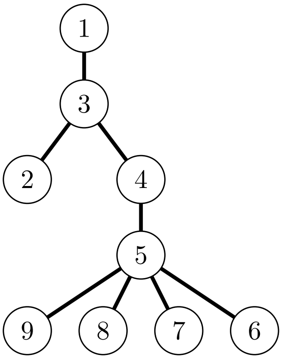
(bv) dc(T) = 4. Distance-consistent vertices: 3, 2, 1, 5



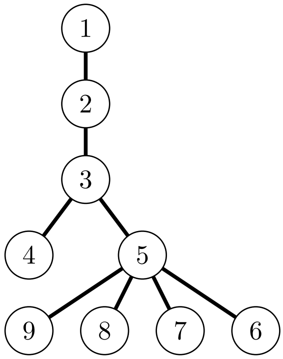
(bw) dc(T) = 4. Distance-consistent vertices: 3, 2, 1, 5



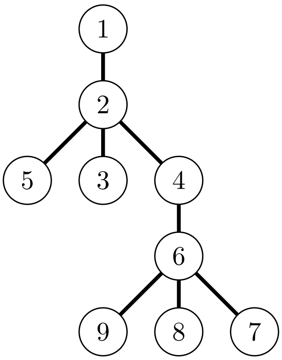
(bx) dc(T) = 3. Distance-consistent vertices: 3, 2, 4



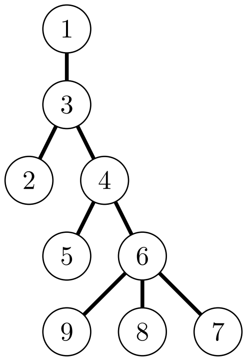
(by) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 4, 6, 3, 2



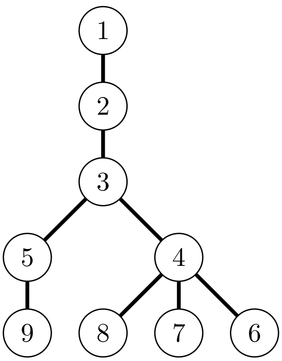
(bz) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 3, 2, 1, 4



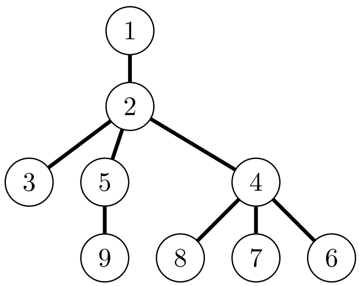
(ca) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



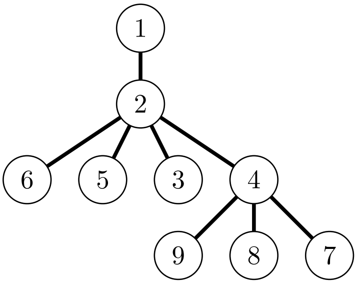
(cb) $\mathrm{dc}(T) = 4$. Distance-consistent vertices: 4, 3, 2, 5



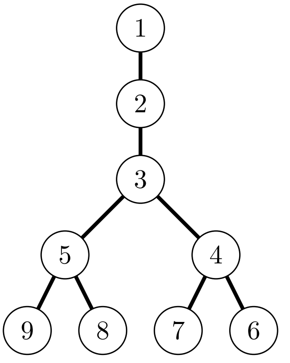
(cc) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 3, 2, 1



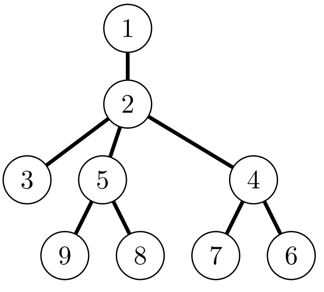
(cd) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



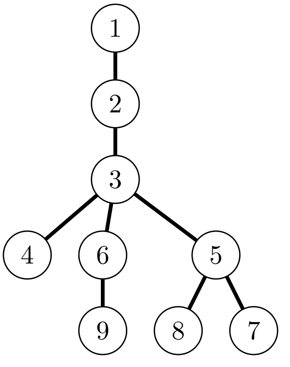
(ce) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



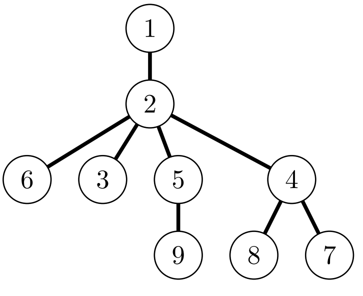
(cf) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 3, 2, 1



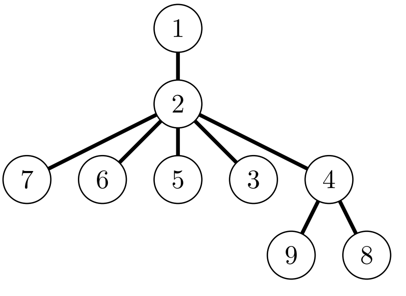
(cg) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



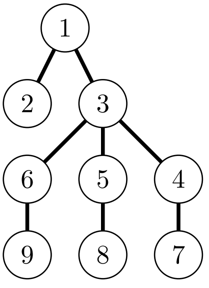
(ch) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 4



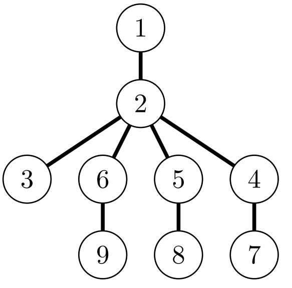
(ci) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



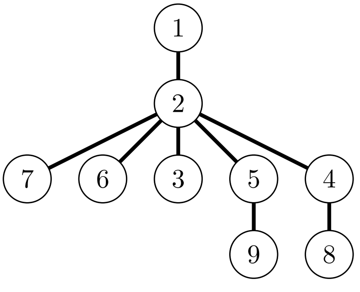
(cj) $\mathrm{dc}(T) = 3$. Distance-consistent vertices: 2, 1, 3



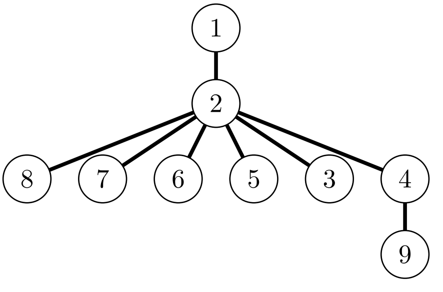
(ck) dc(T) = 2. Distance-consistent vertices: 1, 2



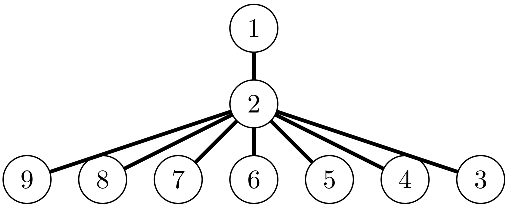
(cl) dc(T) = 3. Distance-consistent vertices: 2, 1, 3



(cm) dc(T) = 3. Distance-consistent vertices: 2, 1, 3



(cn) dc(T) = 3. Distance-consistent vertices: 2, 1, 3



(co) dc(T) = 3. Distance-consistent vertices: 2, 1, 3

Linköping University Electronic Press