离散数学

Discrete Mathematics

for Computer Science

计算机学院计科系

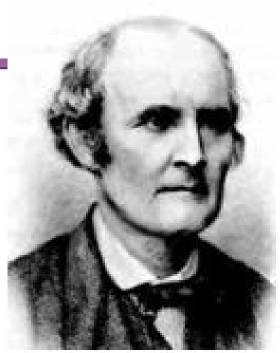
薛思清 xuesiqing@cug.edu.cn

第16讲 树 Tree (1)

I think that I shall never see
A graph more lovely than a tree.

by Radia Perlman

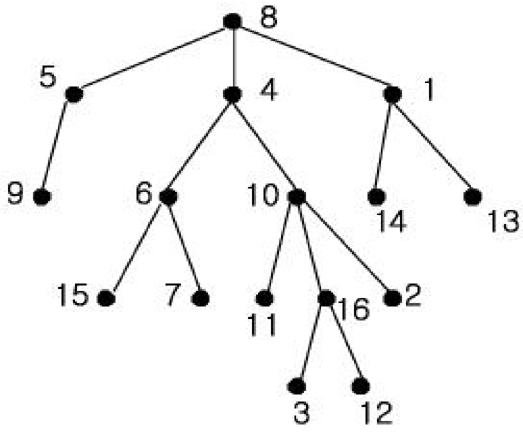
Outline



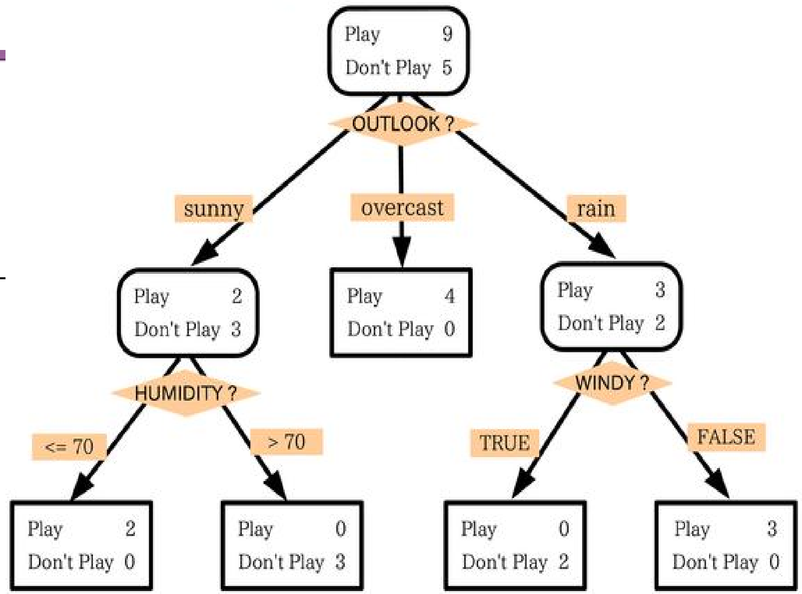
凯莱(Arthur Cayley)
(1821-1895)

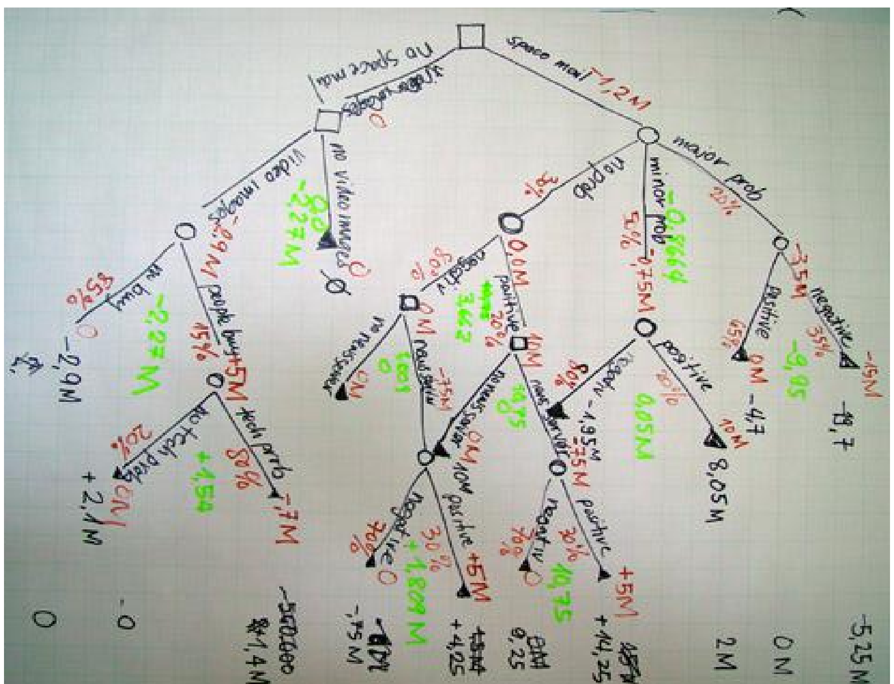
树及其基本性质

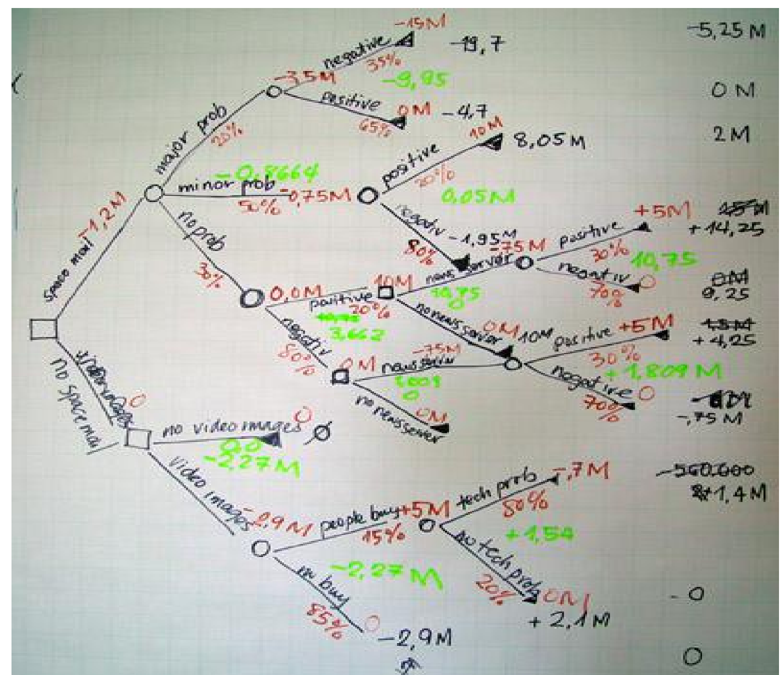
算法理论分析



Dependent variable: PLAY



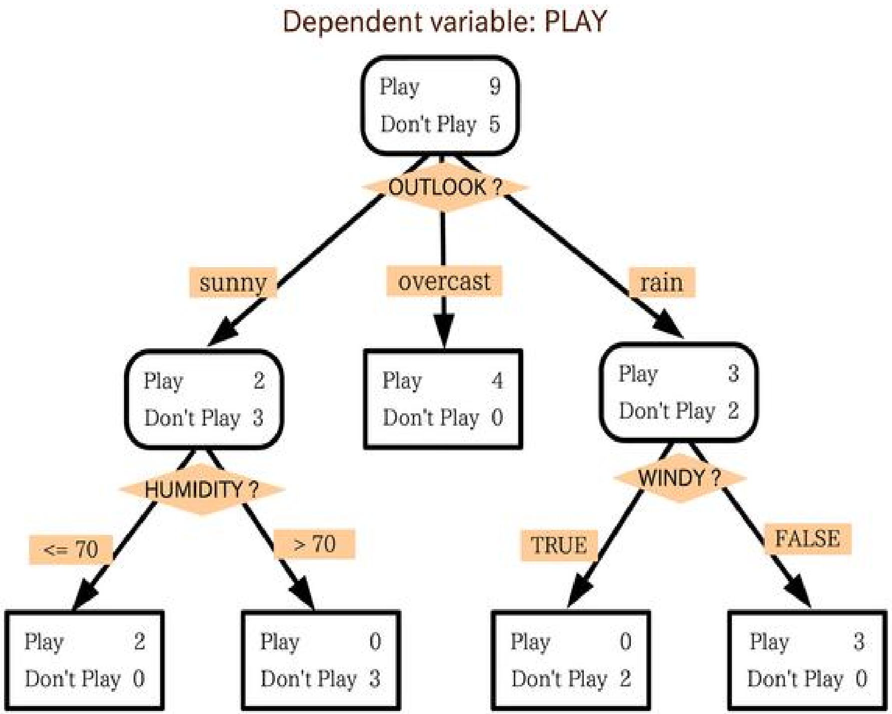




Decision tree 决策树

Play golf dataset

Independent variables				Dep. var
OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	FALSE	Don't Play
sunny	80	90	TRUE	Don't Play
overcast	83	78	FALSE	Play
rain	70	96	FALSE	Play
rain	68	80	FALSE	Play
rain	65	70	TRUE	Don't Play
overcast	64	65	TRUE	Play
sunny	72	95	FALSE	Don't Play
sunny	69	70	FALSE	Play
rain	75	80	FALSE	Play
sunny	75	70	TRUE	Play
overcast	72	90	TRUE	Play
overcast	81	75	FALSE	Play
rain	71	80	TRUE	Don't Play



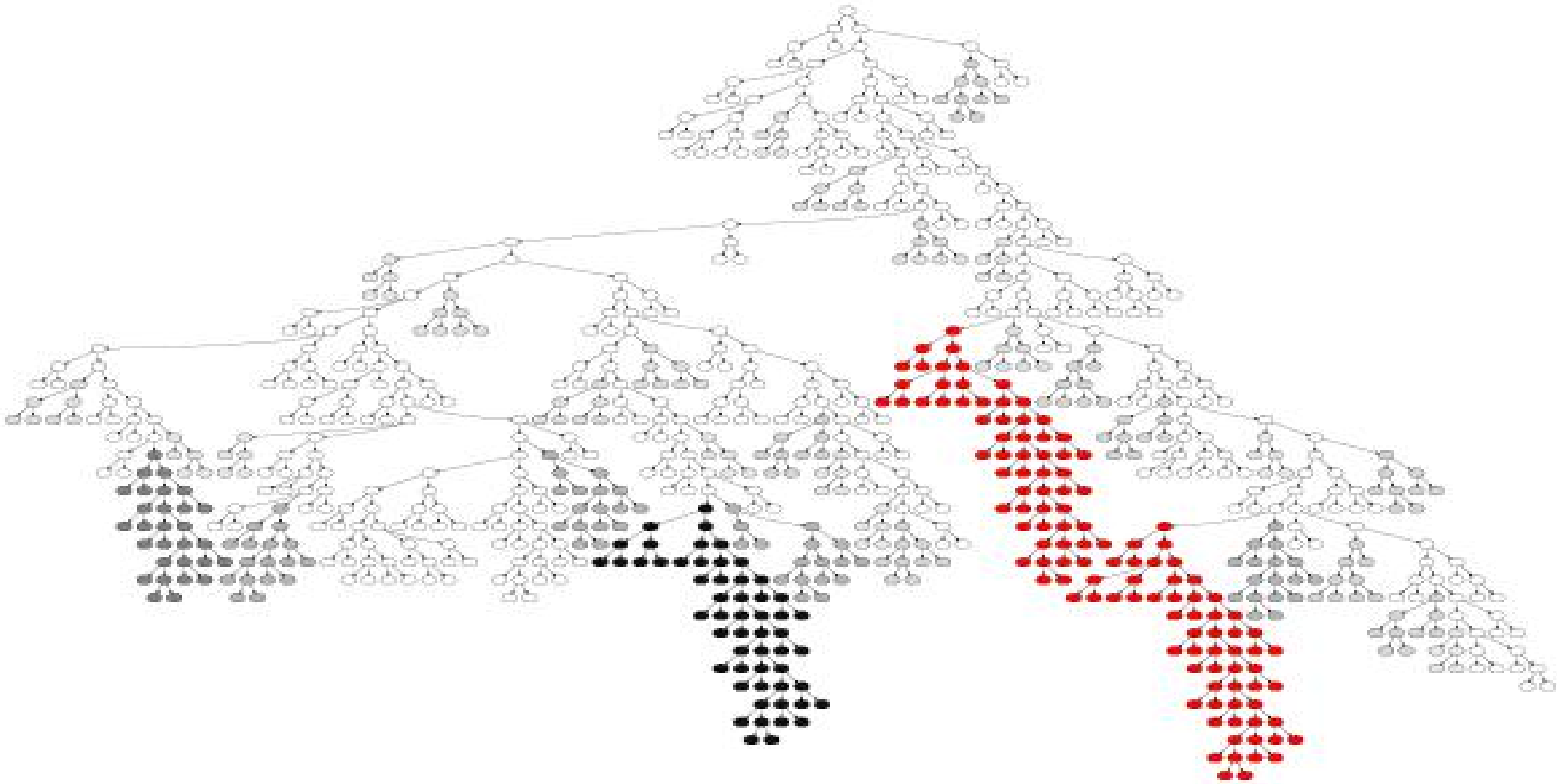
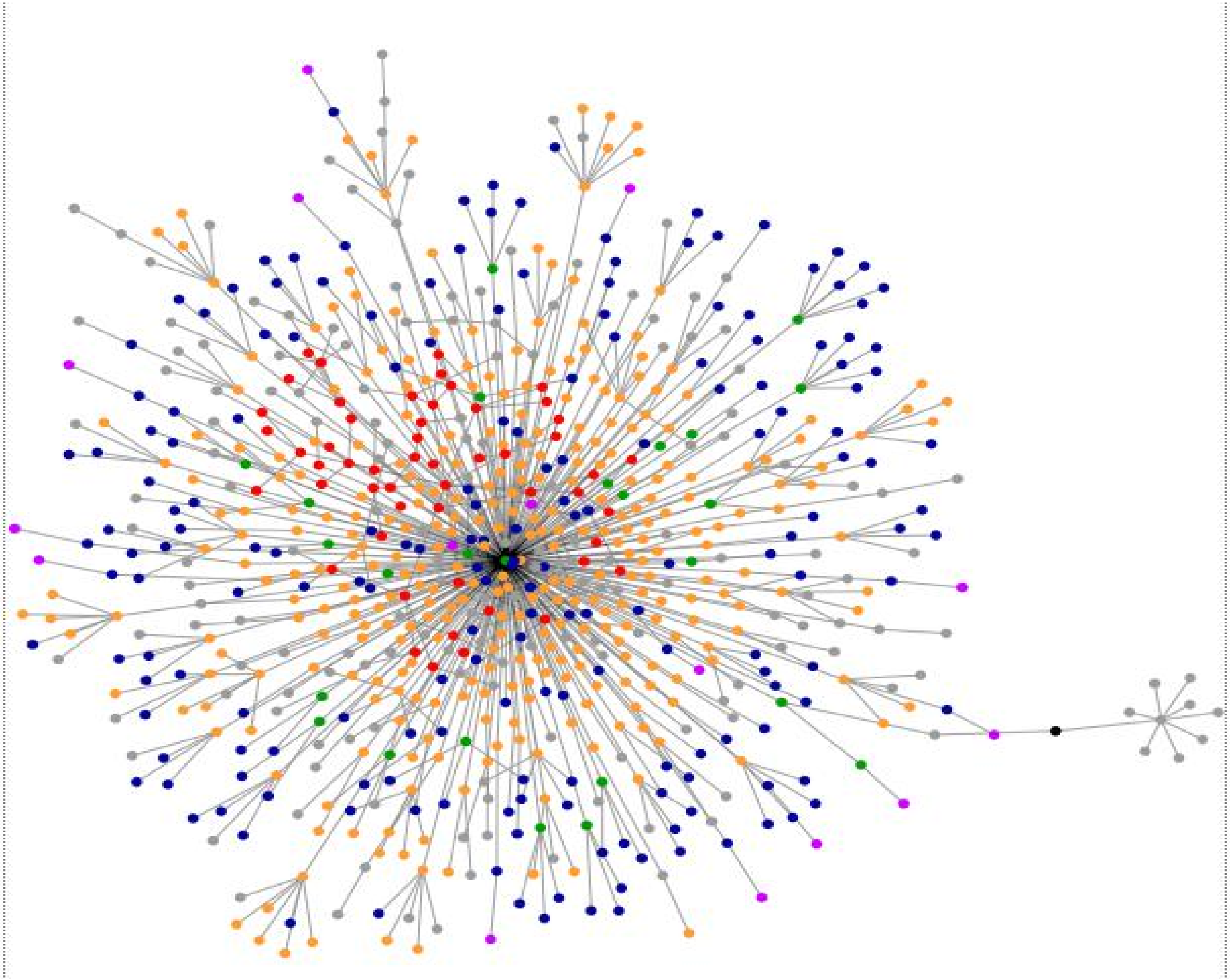
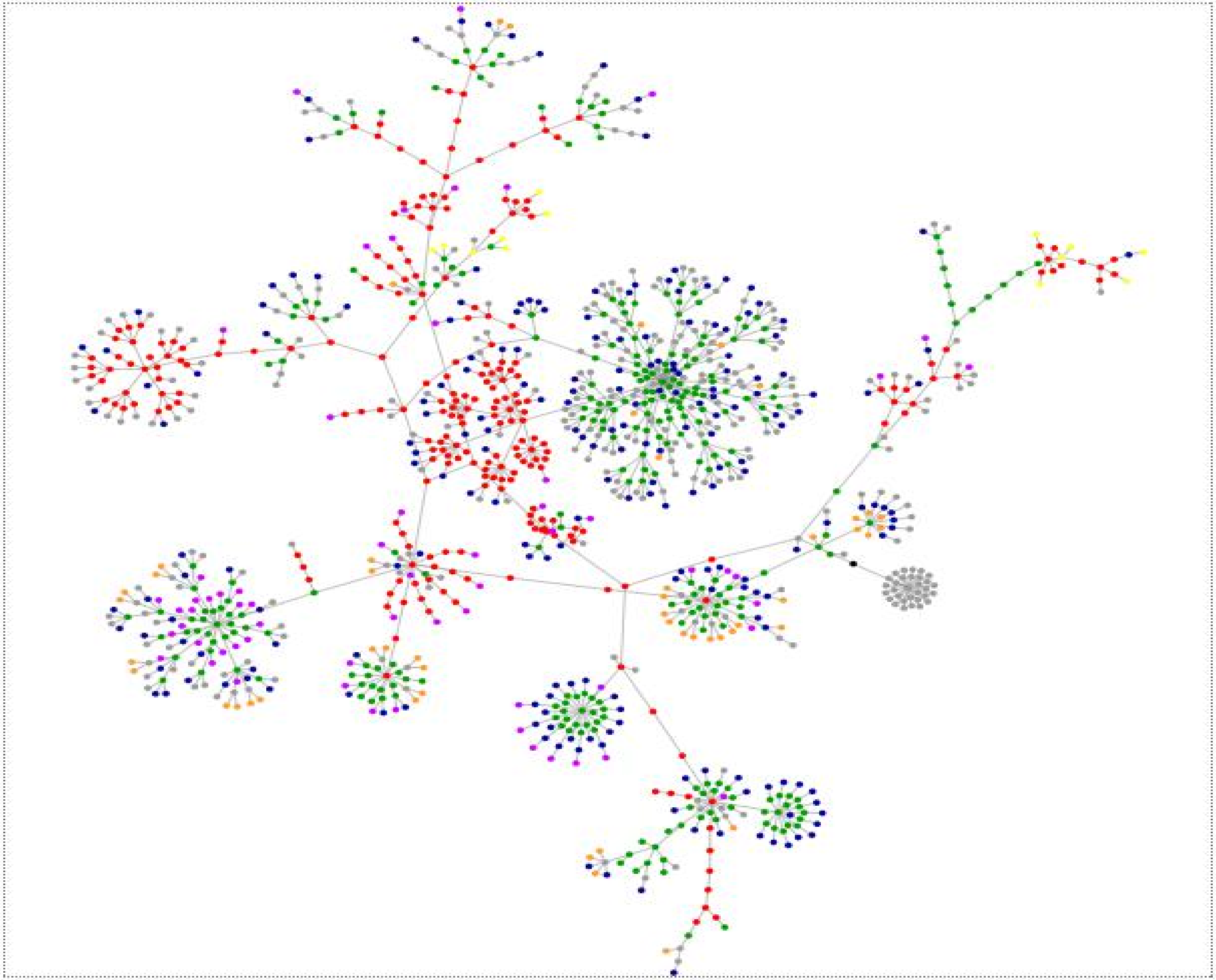
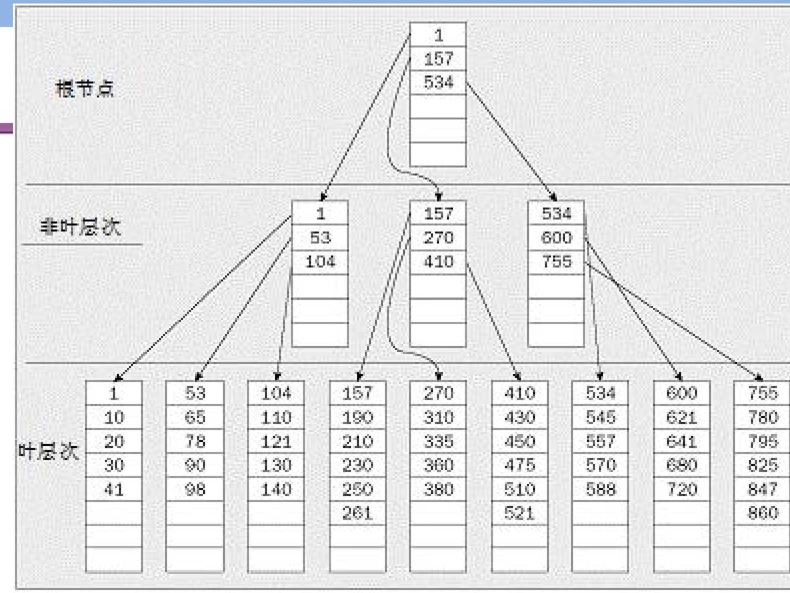


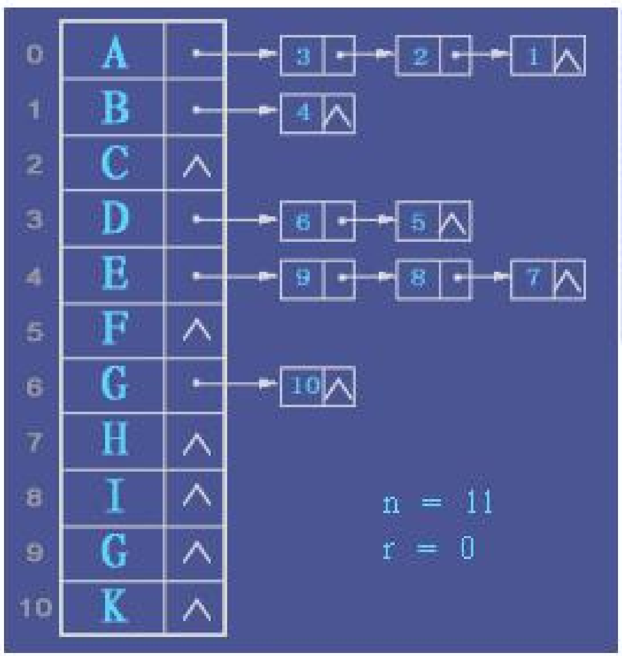
Fig. 6. Same program as in Figure 5. Here whole subtrees are exactly repeated. Nodes are filled according to size of the repeated subtree. Unique nodes and nodes which are part of small patterns (3 nodes or less) are not filled. Two largest (59 nodes, right hand side) coloured red. Note these are partially repeated elsewhere in the tree (e.g. 55 node subtree shaded black).

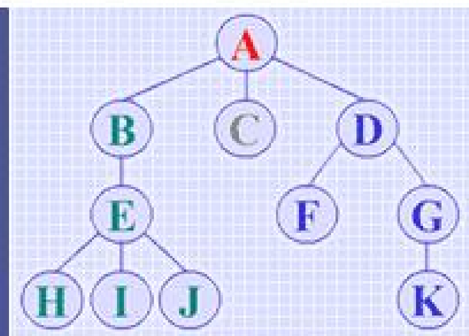




```
/*树的孩子链表存储表示*/
typedef struct CTNode { // 孩子结点
  int child;
  struct CTNode *next;
} *ChildPtr;
typedef struct {
  ElemType data;  // 结点的数据元素
  ChildPtr firstchild;  // 孩子链表头指针
} CTBox;
typedef struct {
  CTBox nodes[MAX_TREE_SIZE];
  int n, r;  // 结点数和根结点的位置
} CTree;
```







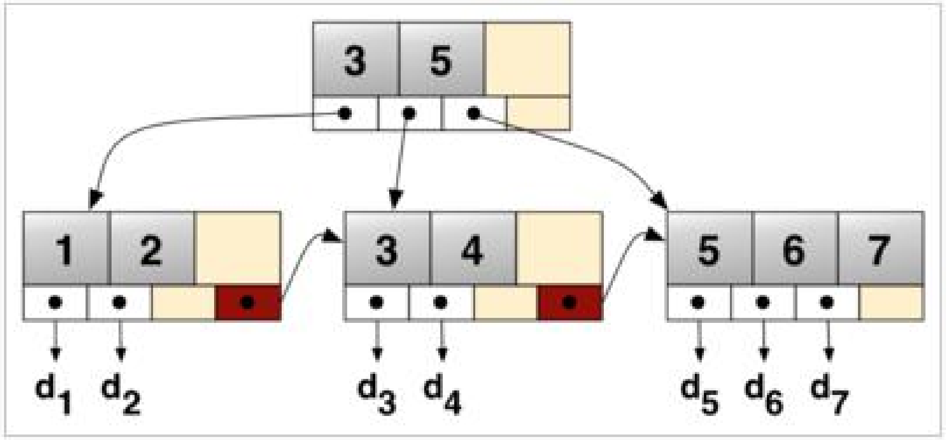
B+ tree

From Wikipedia, the free encyclopedia

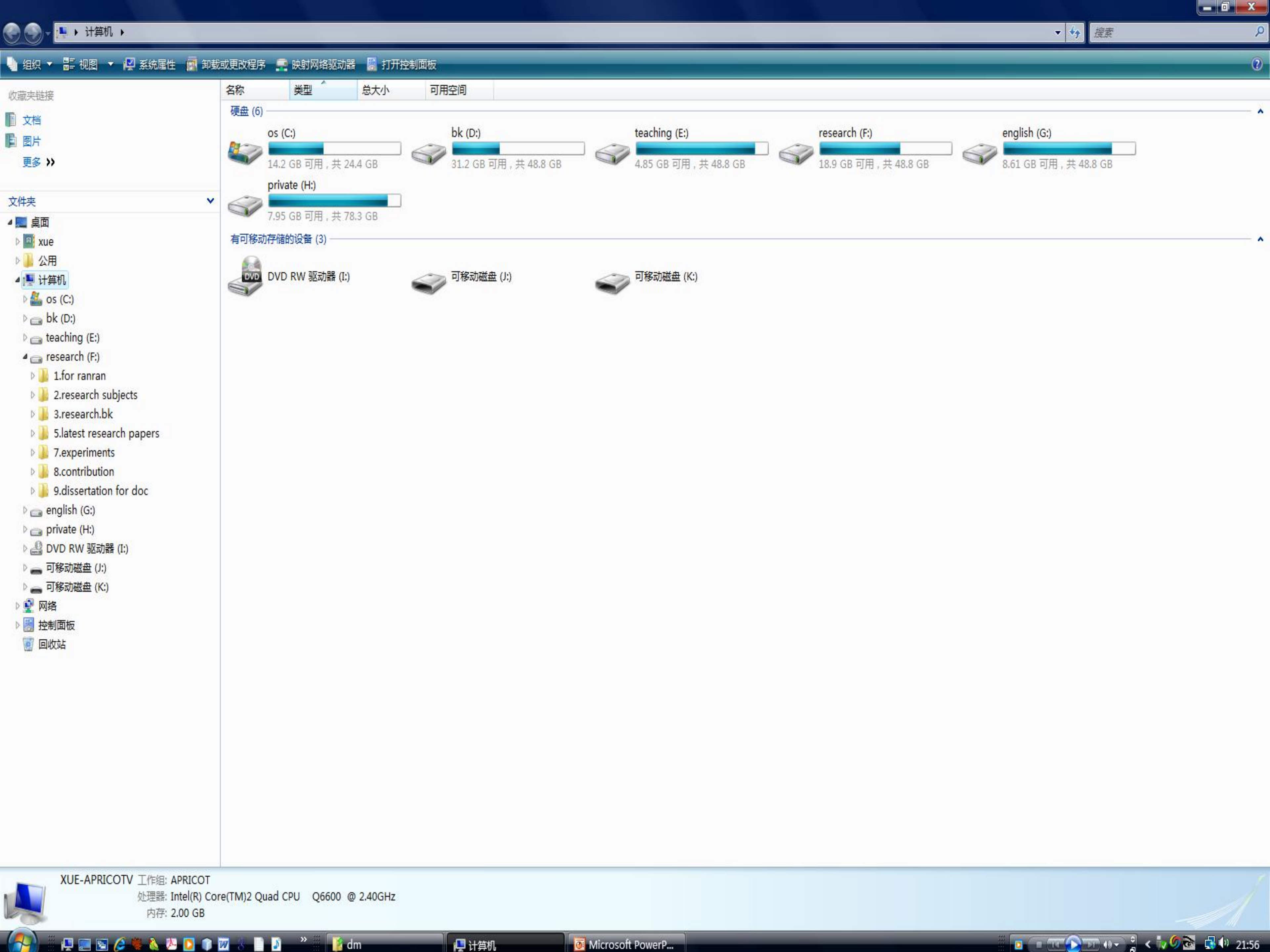
In computer science, a **B+ tree** or **B plus tree** is a type of tree which represents sorted data in a way that allows for efficient insertion, retrieval and removal of records, each of which is identified by a *key*. It is a dynamic, multilevel index, with maximum and minimum bounds on the number of keys in each index segment (usually called a "block" or "node"). In a B+ tree, in contrast to a B-tree, all records are stored at the leaf level of the tree; only keys are stored in interior nodes.

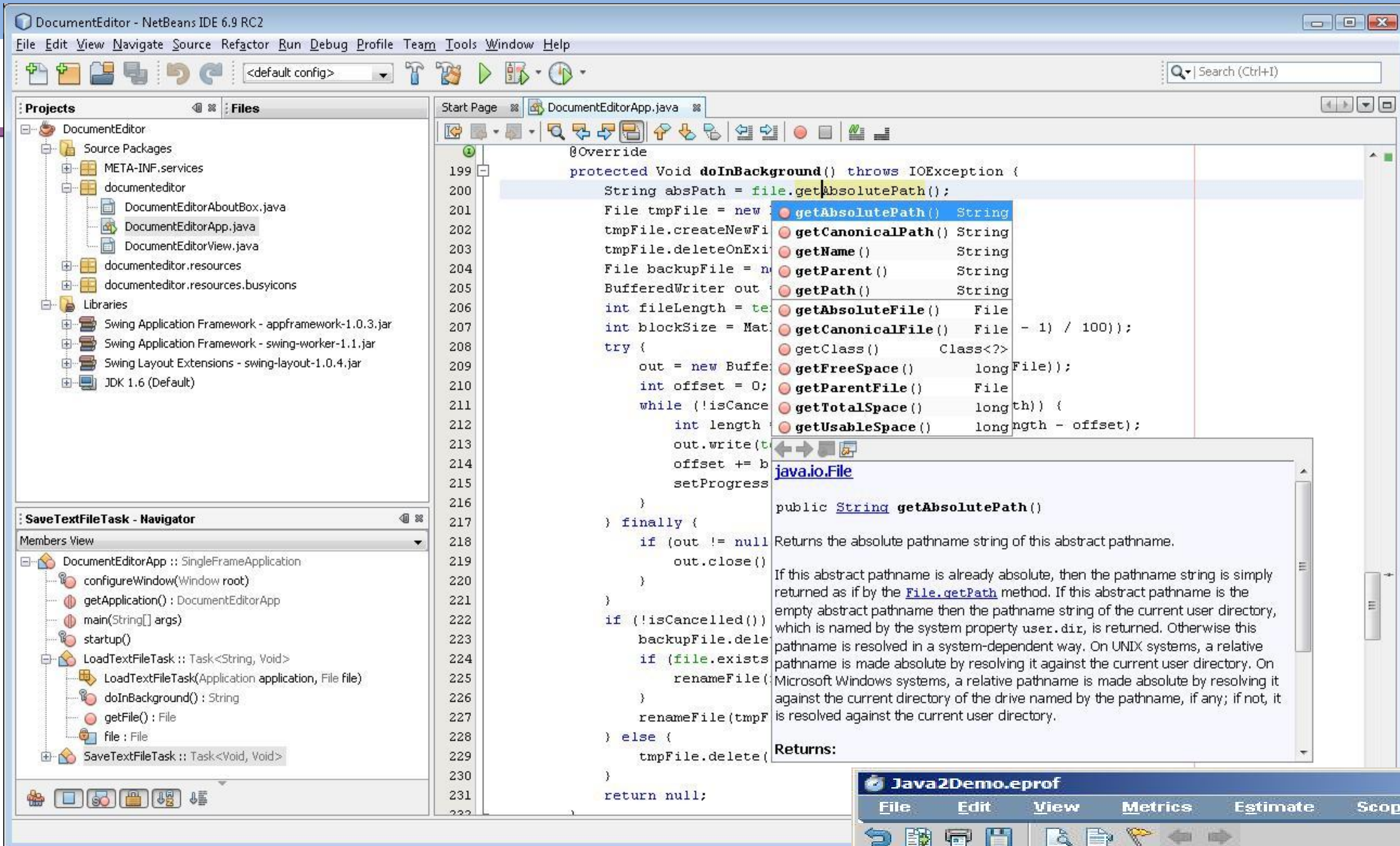
The primary value of a B+ tree is in storing data for efficient retrieval in a block-oriented storage context—in particular, filesystems. This is primarily because unlike binary search trees, B+ trees have very high fanout (typically on the order of 100 or more), which reduces the number of I/O operations required to find an element in the tree.

NTFS, ReiserFS, NSS, XFS, and JFS filesystems all use this type of tree for metadata indexing. Relational database management systems such as IBM DB2[1], Informix[1], Microsoft SQL Server[1], Oracle 8[1], Sybase ASE[1], PostgreSQL[2], Firebird, MySQL[3] and SQLite[4] support this type of tree for table indices. Key-value database management systems such as CouchDB[5], Tokyo Cabinet[6] and Tokyo Tyrant support this type of tree for data access. InfinityDB[7] is a concurrent BTree.

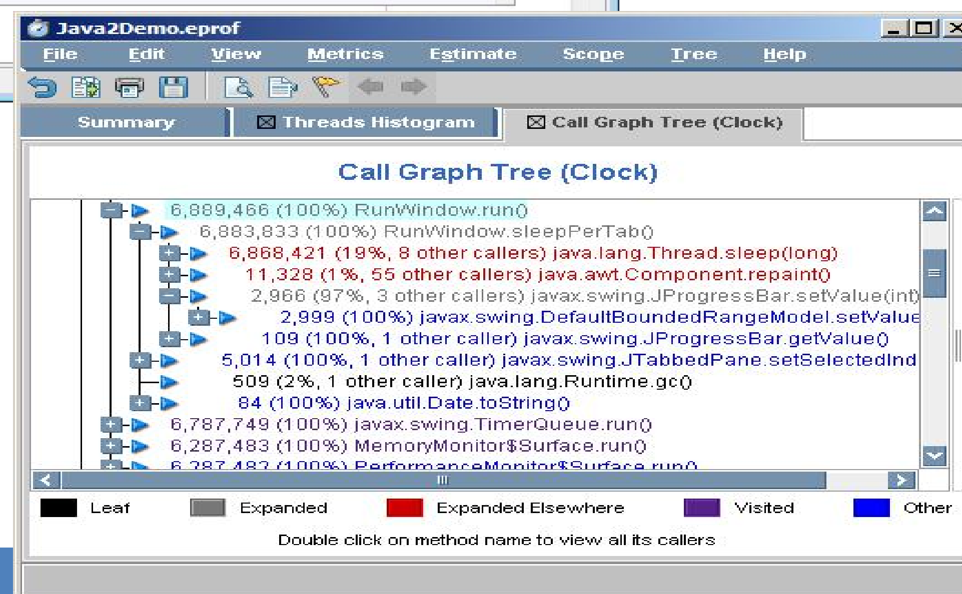


A simple B+ tree example linking the keys 1–7 to data values d_1–d_7. The linked list (red) allows rapid in-order traversal.


计算机
搜索
组织 视图 系统属性 卸载或更改程序 映射网络驱动器 打开控制面板
收藏夹链接
文档
图片
更多 »
名称 类型 总大小 可用空间
硬盘 (6)
os (C:) 14.2 GB 可用，共 24.4 GB
bk (D:) 31.2 GB 可用，共 48.8 GB
teaching (E:) 4.85 GB 可用，共 48.8 GB
research (F:) 18.9 GB 可用，共 48.8 GB
english (G:) 8.61 GB 可用，共 48.8 GB
private (H:) 7.95 GB 可用，共 78.3 GB
有可移动存储的设备 (3)
DVD RW 驱动器 (I:)
可移动磁盘 (J:)
可移动磁盘 (K:)
文件夹
桌面
xue
公用
计算机
os (C:)
bk (D:)
teaching (E:)
research (F:)
1.for ranran
2.research subjects
3.research.bk
5.latest research papers
7.experiments
8.contribution
9.dissertation for doc
english (G:)
private (H:)
DVD RW 驱动器 (I:)
可移动磁盘 (J:)
可移动磁盘 (K:)
网络
控制面板
回收站
XUE-APRICOTV 工作组: APRICOT
处理器: Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz
内存: 2.00 GB
dm
计算机
Microsoft PowerP...
21:56



DocumentEditor - NetBeans IDE 6.9 RC2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Files
DocumentEditor
Source Packages
META-INF.services
documenteditor
DocumentEditorAboutBox.java
DocumentEditorApp.java
DocumentEditorView.java
documenteditor.resources
documenteditor.resources.busyicons
Libraries
Swing Application Framework - appframework-1.0.3.jar
Swing Application Framework - swing-worker-1.1.jar
Swing Layout Extensions - swing-layout-1.0.4.jar
JDK 1.6 (Default)
SaveTextFileTask - Navigator
Members View
DocumentEditorApp :: SingleFrameApplication
LoadTextFileTask :: Task<String, Void>
SaveTextFileTask :: Task<Void, Void>
Start Page
DocumentEditorApp.java
protected Void doInBackground() throws IOException {
String absPath = file.getAbsolutePath();
java.io.File
public String getAbsolutePath()
Returns the absolute pathname string of this abstract pathname.
If this abstract pathname is already absolute, then the pathname string is simply returned as if by the File.getPath method. If this abstract pathname is the empty abstract pathname then the pathname string of the current user directory, which is named by the system property user.dir, is returned. Otherwise this pathname is resolved in a system-dependent way. On UNIX systems, a relative pathname is made absolute by resolving it against the current user directory. On Microsoft Windows systems, a relative pathname is made absolute by resolving it against the current directory of the drive named by the pathname, if any; if not, it is resolved against the current user directory.
Returns:



Java2Demo.eprof
File Edit View Metrics Estimate Scope Tree Help
Summary
Threads Histogram
Call Graph Tree (Clock)
Call Graph Tree (Clock)
6,889,466 (100%) RunWindow.run()
6,883,833 (100%) RunWindow.sleepPerTab()
6,868,421 (19%, 8 other callers) java.lang.Thread.sleep(long)
11,328 (1%, 55 other callers) java.awt.Component.repaint()
2,966 (97%, 3 other callers) javax.swing.JProgressBar.setValue(int)
2,999 (100%) javax.swing.DefaultBoundedRangeModel.setValue
109 (100%, 1 other caller) javax.swing.JProgressBar.getValue()
5,014 (100%, 1 other caller) javax.swing.JTabbedPane.setSelectedInd
509 (2%, 1 other caller) java.lang.Runtime.gc()
84 (100%) java.util.Date.toString()
6,787,749 (100%) javax.swing.TimerQueue.run()
6,287,483 (100%) MemoryMonitor$Surface.run()
Leaf
Expanded
Expanded Elsewhere
Visited
Other
Double click on method name to view all its callers

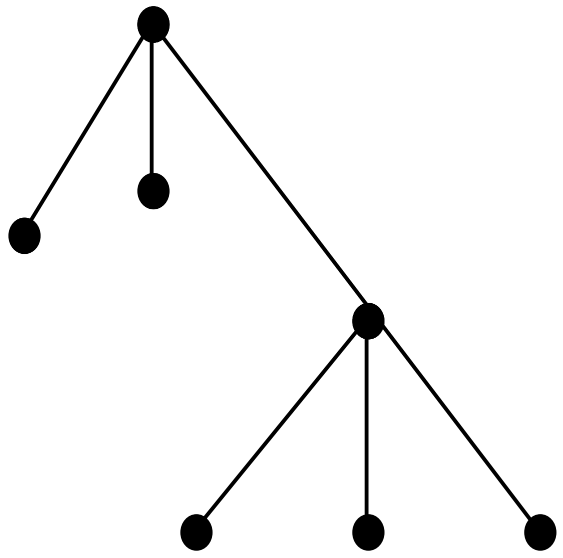
树 (Tree):无环的连通图

森林 (Forest)

平凡树 (Trivial Tree)

树叶 (Leaf)/**外部结点** (External nodes)

分支结点 (Branched Vertex)/**内部结点** (Internal ndes)



性质

设图G=（n,m），如果G满足如下三个属性中的两个，则G就是一棵树，且可以推导出另一个属性：

1) G连通；

2) G中不存在环；

3) m=n-1

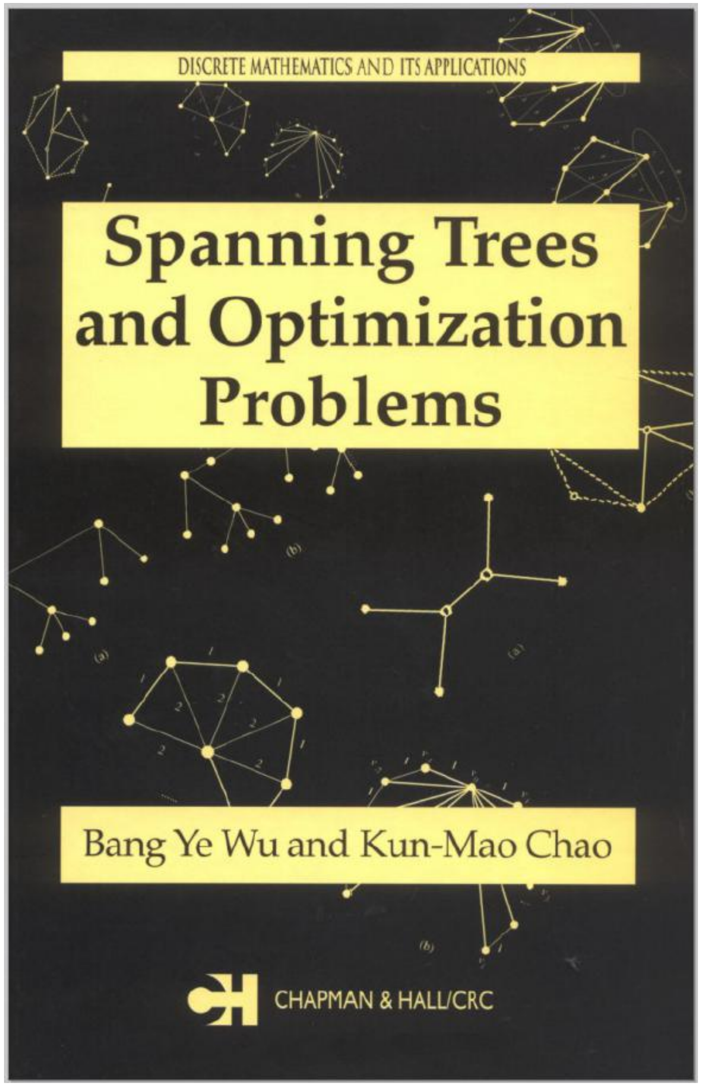
思考 下列命题是否为真?

1) 在树T的任意二结点间添加一条边后，将构成包含该二结点的闭通路，即存在环
2) 删除树T的任意一条边后，T就不连通，即树T的每一条边均为T的割边
3) 树T的每一对结点间存在惟一一条路径
4) 结点数大于等于2的任意树，至少有两片树叶
5) 图G为n个结点、w个分图的森林，则G边数为n-w

示例

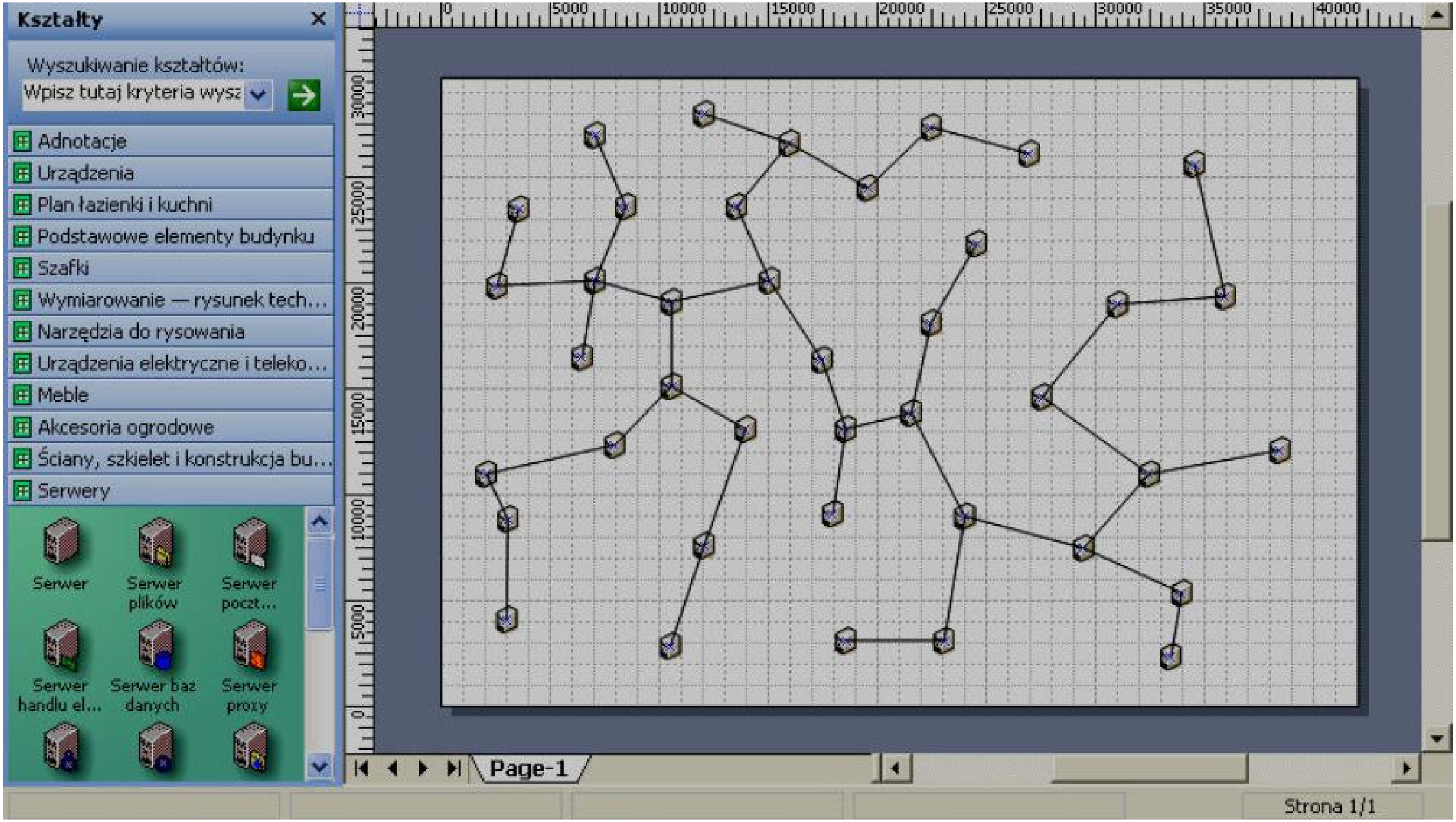
树T具有n_i个度数为i的结点，i=2，3，…,k,其余为叶子结点，问该树有多少叶子结点?

2 生成树 Spanning Tree



This book provides a comprehensive introduction to the modern study of spanning trees. A spanning tree for a graph G is a subgraph of G that is a tree and contains all the vertices of G. There are many situations in which good spanning trees must be found. Whenever one wants to find a simple, cheap, yet efficient way to connect a set of terminals, be they computers, telephones, factories, or cities, a solution is normally one kind of spanning trees. Spanning trees prove important for several reasons:

1. They create a sparse subgraph that reflects a lot about the original graph.
2. They play an important role in designing efficient routing algorithms.
3. Some computationally hard problems, such as the Steiner tree problem and the traveling salesperson problem, can be solved approximately by using spanning trees.
4. They have wide applications in many areas, such as network design, bioinformatics, etc.


Kształty
Wyszukiwanie kształtów:
Wpisz tutaj kryteria wysz
Adnotacje
Urządzenia
Plan łazienki i kuchni
Podstawowe elementy budynku
Szafki
Wymiarowanie — rysunek tech...
Narzędzia do rysowania
Urządzenia elektryczne i teleko...
Meble
Akcesoria ogrodowe
Ściany, szkielet i konstrukcja bu...
Serwery
Serwer
Serwer plików
Serwer poczt...
Serwer handlu el...
Serwer baz danych
Serwer proxy
Page-1
Strona 1/1

Evolutionary Approach to Constrained Minimum Spanning Tree Problem – Commercial Software Based Application

Anna Pagacz [1], Günther Raidl [2], Stanisław Zawiślak [3]

[1] Vienna University of Technology, Socrates student 2004/2005; regular since 2005-onwards, Vienna, Austria, e-mail: annapagacz@poczta.onet.pl.
[2] Vienna University of Technology, Institute of Computer Graphics and Algorithms, Vienna, Austria, e-mail: raidl@ads.tuwien.ac.at,
[3] University of Bielsko-Biała, Faculty of Mechanical Engineering and Computer Science, Bielsko-Biała, Poland, e-mail : szawislak@ath.bielsko.pl.

Abstract. The constrained minimum spanning tree problem is considered in the paper. We assume that the degree of any vertex should not exceed a particular constraint d. In this formulation the problem turns into NP-hard one, therefore the evolutionary approach is applicable. The edge set representation of a chromosome was utilized for a tree in the algorithm. The evolutionary algorithm was worked out and the related computer program has been written. Interfaces between the core program and MS Visio as well as the data base system were prepared. The results obtained by means of the system are shown.

1 Introduction

Graph theory problems are solved by means of versatile algorithms which have been developed since the origins of the theory in the eighteen century. Some NP-hard problems are solved by means of evolutionary algorithms.

The goal of this paper is to give a short review of the papers dealing with algorithms for generalized spanning tree problems i.e. in non-classical formulated versions as well as to describe a computer program system (using commercial software) developed to solve this problem for one particular formulation.

It is a well known fact that an exact solution can be found for some classical problems in plain formulations e.g. minimum spanning tree problem (MST-P) or the shortest path problem. However, even those are turned into NP-hard problems when constraints are introduced or some additional assumptions are made. For example, it has been proved that quadratic minimum spanning tree (q-MST) is NP-hard [18]. However an EA approach is reasonable because of their multi-objective formulation. This area is especially suitable for EA because in the multi-objective approach we consider a Pareto set of solutions which can be analyzed iteration by iteration considering the whole population of solutions instead of one solution [4]. Furthermore, it has been proven that an evolutionary approach to some graph theory problems is comparably more effective than some other approaches e.g. ant colonies, neuronal networks and other approaches belonging to AI domain. Moreover EA-based methodology is especially flexible, robust and handy [3]. At present, telecommunication is one of a very fast growing field of science. At present, telecommunication presents a challenging range of difficult design

problem formulation

Let's consider an undirected complete graph G = (V, E) where V = $\{v_1, \ldots, v_n\}$ is the set of n *nodes* (or *vertices*) and E = $\{e_1, \ldots, e_m\}$ is the set of m *arcs* (or *edges*) with given costs c_{ij} for each arc e $\in$ E, connecting vertices v_i and v_j. The *degree constrained minimum spanning tree* (DCMST) problem on G is to find a spanning tree of minimum total cost, such that the degree of each node is at most a given value d_i. Degree is defined as the number of arcs incident to v_i. For simplicity v_i is denoted by i in figures and some descriptions.

$$c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad (1)$$

Because a network can be represented by a graph, in which each vertex represents a single node, therefore it is necessary to minimize the objective function:

$$C = \sum_{(i,j)} c_{ij} \longrightarrow \min \qquad (2)$$

It means, we have to find a graph with the minimal total cost, in other words we have to find a minimum spanning tree that contains all network's devices, which are laid out by user in MS Visio.

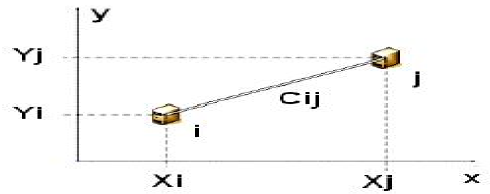


Figure 1. Calculating the weight c_{ij} of an edge(i, j); part of MS Visio screen editor

We wanted to find a minimal total cost of connection between all nodes, for the given layout of devices, and taking into account degree-constraint condition. Given constrains are as follows:

$$C = \sum_{(i,j)} c_{ij} \longrightarrow \min$$
$$\deg(i) \le \mathbf{d}, \quad i \in V. \qquad (3)$$

For this formulation of the problem, the computer program was created and tested for up to 60-vertex-graphs.

An Ant Colony Optimization Approach for Degree-constrained Minimum Spanning Tree Problem

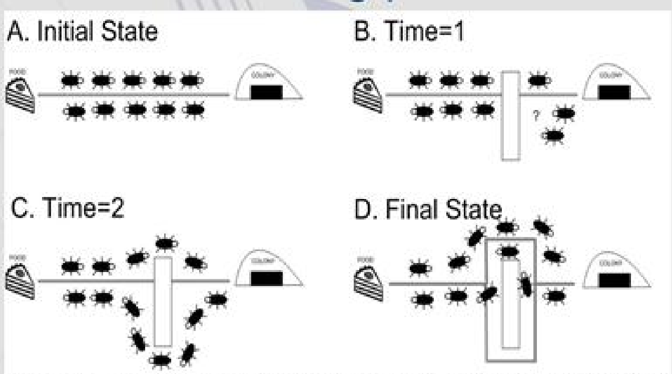
*
By

Yoon-Teck Bau, Chin-Kuan Ho, Hong-Tat Ewe
Faculty of Information Technology
Multimedia University, Malaysia
{ytbau, ckho, htewe}@ @mmu.edu.my
http://pesona.mmu.edu.my/~ytbau/

My Research Outline

- ACO the big picture
- Brief overview of ACO
- Scope of work
- Degree-constrained minimum spanning tree (d-MST) problem formulation
- Simple Problem and Solution
- Visualisation Experiment
- d-MST application
- Literature research
- The proposed p-ACO approach
- Summary and future work
- References

ACO the big picture



Figures above show **A.** Ants in a pheromone trail between colony and food; **B.** an obstacle interrupts the trail; **C.** ants find two paths to go around the obstacle; **D.** a new pheromone trail is formed along the shorter path. Ants converging on the shortest path in the graph from the colony to a food source and back.

Brief overview of ACO

- Ants follow the principle of stigmergy (Grasse',1959).
- Stigmergy is a form of indirect communication between agents which have effects upon the environment, which serve as behavior-determining signals to other agents that is that pheromone trails.
- ACO Metaheuristic is a high-level strategy that guides other heuristics in a search for feasible solutions to solve optimization problems.

A New Ant Colony Optimization Approach for the Degree-Constrained Minimum Spanning Tree Problem Using Prüfer and Blob Codes Tree Coding

Yoon-Teck Bau, Chin-Kuan Ho and Hong-Tat Ewe
Faculty of Information Technology, Multimedia University
Malaysia

1. Introduction

This chapter describes a novel ACO algorithm for the degree-constrained minimum spanning tree (d-MST) problem. Instead of constructing the d-MST directly on the construction graph, ants construct the encoded d-MST. Two well-known tree codings are used: the Prüfer code, and the more recent Blob code (Picciotto, 1999). Both of these tree codings are bijective because they represent each spanning tree of the complete graph on $|V|$ labelled vertices as a code of $|V|$-2 vertex labels. Each spanning tree corresponds to a unique code, and each code corresponds to a unique spanning tree. Under the proposed approach, ants will select graph vertices and place them into the Prüfer code or Blob code being constructed. The use of tree codings such as Prüfer code or Blob code makes it easier for the proposed ACO to solve another variant of the d-MST problem with both lower and upper bound constraints on each vertex (lu-dMST). A general lu-dMST problem formulation is given. This general lu-dMST problem formulation could be used to denote d-MST problem formulation also. Subsequently, Prüfer code and Blob code tree encoding and decoding are presented and then followed by the design of two ACO approaches using these tree codings to solve d-MST and lu-dMST problems. Next, results from these ACO approaches are compared on structured hard (SHRD) graph data set for both d-MST and lu-dMST problems, and important findings are reported.

2. Problem Formulation

In this chapter, a special case of degree-constrained minimum spanning tree where the lower and upper bound of the number of edges is imposed on each vertex is considered. This similar to the problem being solved by Chou et al. (2001), and is named lu-dMST in this chapter. Chou et al. (2001) named this problem as DCMST. The d-MST problem is different since it has only the upper bound constraint. Chou et al. (2001) also proposed the following notation to be used for the lu-dMST problem formulation:

$G = (V, E)$ connected weighted undirected graph.

i, j = index of labelled vertices $i, j = 0, 1, 2, \ldots, |V-1|$.

Open Access Database www.i-techonline.com

Source: Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, Book edited by: Felix T. S. Chan and Manoj Kumar Tiwari, ISBN 978-3-902613-09-7, pp. 532, December 2007, Itech Education and Publishing, Vienna, Austria

ACM DL DIGITAL LIBRARY

SIGN IN SIGN UP

SEARCH

Min-degree constrained minimum spanning tree problem: New formulation via Miller-Tucker-Zemlin constraints

Authors: İbrahim Akgün Department of Industrial Engineering, Bilkent University, Bilkent 06800, Ankara, Turkey
Barbaros Ç. Tansel Department of Industrial Engineering, Bilkent University, Bilkent 06800, Ankara, Turkey

2010 Article

Published in:
· Journal
Computers and Operations Research archive
Volume 37 Issue 1, January, 2010 table of contents doi>10.1016/j.cor.2009.03.006

Bibliometrics
· Downloads (6 Weeks): n/a
· Downloads (12 Months): n/a
· Citation Count: 1

Tools and Resources

TOC Service:
Email RSS

Save to Binder

Export Formats:
BibTeX EndNote ACM Ref

Share:

Tags: degree-enforcing constraints flow formulation miller-tucker-zemlin constraints more tags

Feedback | Switch to single page view (no tabs)

Abstract | Authors | References | Cited By | Index Terms | Publication | Reviews | Comments | Table of Contents

Given an undirected network with positive edge costs and a positive integer d>2, the minimum-degree constrained minimum spanning tree problem is the problem of finding a spanning tree with minimum total cost such that each non-leaf node in the tree has a degree of at least d. This problem is new to the literature while the related problem with upper bound constraints on degrees is well studied. Mixed-integer programs proposed for either type of problem is composed, in general, of a tree-defining part and a degree-enforcing part. In our formulation of the minimum-degree constrained minimum spanning tree problem, the tree-defining part is based on the Miller-Tucker-Zemlin constraints while the only earlier paper available in the literature on this problem uses single and multi-commodity flow-based formulations that are well studied for the case of upper degree constraints. We propose a new set of constraints for the degree-enforcing part that lead to significantly better solution times than earlier approaches when used in conjunction with Miller-Tucker-Zemlin constraints.

Powered by *THE ACM GUIDE TO COMPUTING LITERATURE*

The ACM Digital Library is published by the Association for Computing Machinery. Copyright © 2010 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

一种求解度约束最小生成树问题的优化算法*

王竹荣+, 张九龙, 崔杜武

(西安理工大学 计算机科学与工程学院,陕西 西安 710048)

Optimization Algorithm for Solving Degree-Constrained Minimum Spanning Tree Problem

WANG Zhu-Rong+, ZHANG Jiu-Long, CUI Du-Wu

(School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China)

+ Corresponding author: E-mail: wangzhurong@xaut.edu.cn

Wang ZR, Zhang JL, Cui DW. Optimization algorithm for solving degree-constrained minimum spanning tree problem. *Journal of Software*, 2010,21(12):3068−3081. http://www.jos.org.cn/1000-9825/3713.htm

Abstract: To solve the degree-constrained spanning minimum tree (DCMST) problems with a large scale of nodes, an optimization algorithm based on grafting and pruning operator is proposed. Learning from the flower planting techniques, this paper establishes, an evolutionary computation framework containing accelerating and adjusting operators based on conventional genetic operators. The grafting and pruning are performed by a greedy strategy and gain maximization respectively. The collision caused by possible local minima is analyzed and detected, and several methods dealing with the collision are discussed. To tackle the complexity of DCMST problems, some strategies of grafting and pruning are proposed. The convergence of the proposed algorithm and the computation complexity are analyzed. For DCMST problems of Euclidean and uniform random non-Euclidean instances from 50 to 500 nodes, the experiments show that the quality and convergence rate of the proposed method are the best compared with the known results.

Key words: DCMST; genetic algorithm; grafting; pruning

摘 要: 为求解大规模结点度约束最小生成树问题,提出一种带有嫁接和剪接算子操作的优化算法.通过借鉴花草果树种植技术,建立一种以基本遗传算子为基础、带有加速和调节算子作为激励的进化计算体系;嫁接以一种贪婪的思想加速搜索,按收益最大化原则进行剪接.对可能陷入局部极值引起冲突的现象及冲突检测的方法进行分析,并提出了冲突的若干解决方法.针对 DCMST 问题求解中的复杂性,提出了几种有效的嫁接和剪接的策略,并对算法的收敛性和计算复杂度进行了分析.通过该算法对结点数为 50~500 之间的 Euclidean 问题和按均匀随机方式产生的 non-Euclidean 度约束最小生成树问题进行求解.与现有文献的实验结果对比表明,该方法在求解最好解的精度和收敛速度上均有一定的优势.

关键词: 度约束最小生成树;遗传算法;嫁接;剪接

中图法分类号: TP301 文献标识码: A

www.jos.org.cn

Spanni

From Wikipedi

The **Spannin** ... res a
loop-free to ... revent
bridge loops

In the OSI m ... s
standardized ... a mesh
network of c ... s those
links that a ... een any
two network

Spanning tre ... vide
automatic ba ... ops, or
the need for ... be
avoided beca

STP is based ... l
Equipment Co

Spanning Tree Protocol

Radia Perlman

I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree which must be sure to span.
So packets can reach every LAN.
First the Root must be selected
By ID it is elected.
Least cost paths from Root are traced
In the tree these paths are placed.
A mesh is made by folks like me
Then bridges find a spanning tree

生成树

若无向图的一个生成子图T是树，则称T为G的生成树

树枝、弦

生成树的构造方法？

——“破圈”法

请你思考

1) 只有连通图才有生成树?

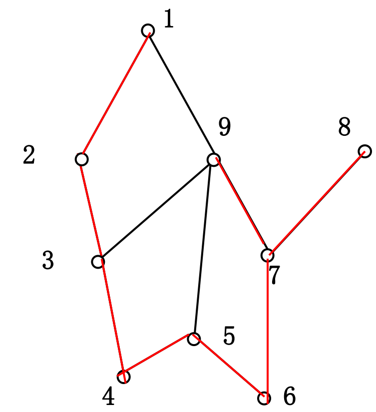
2) 连通图的至少有一棵生成树?

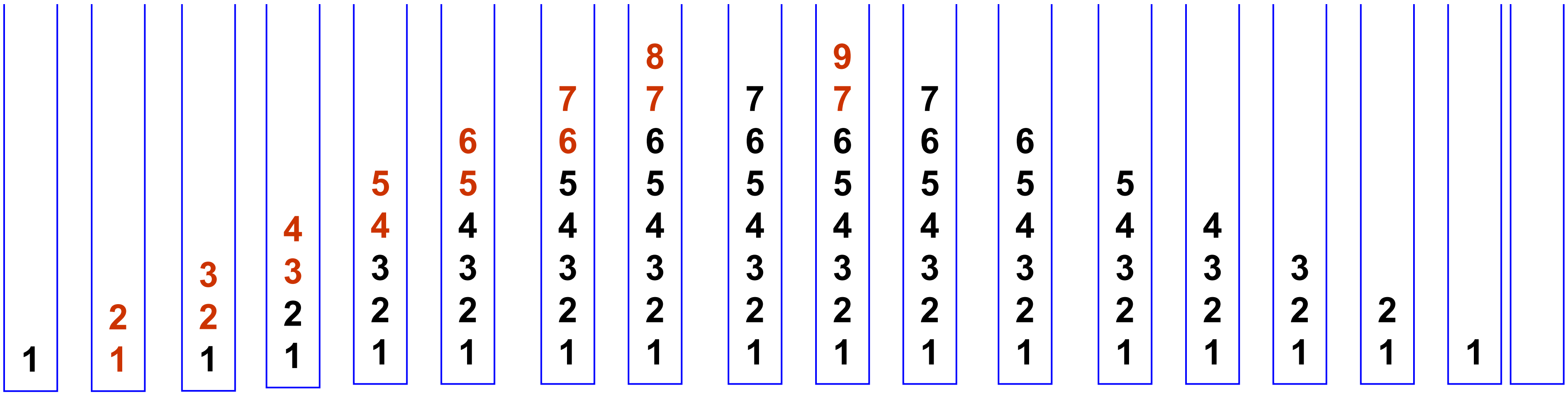
3) 设G为连通无向图，那么G的任一回路与G – T至少有一条公共边.

4) 生成树的求解与数量问题.

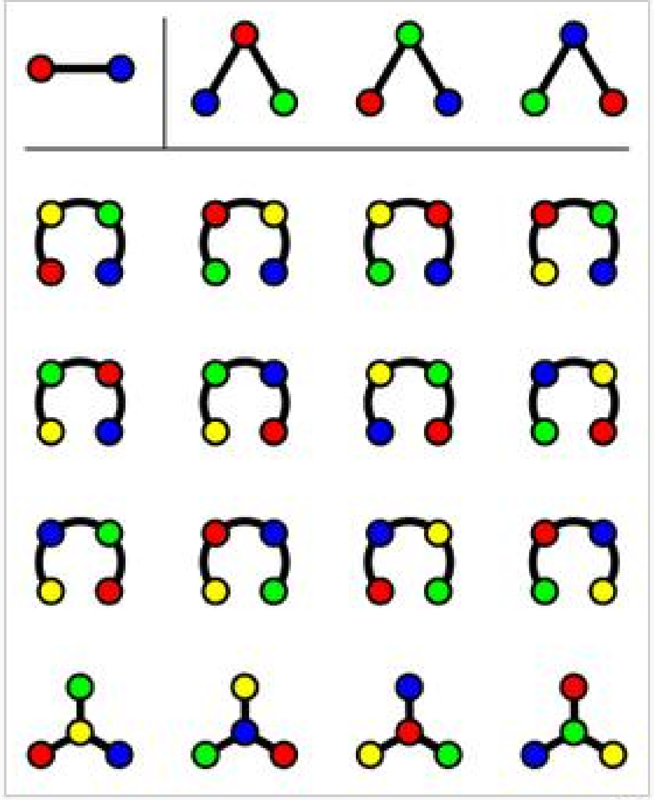
求解生成树算法

(1) DFS算法　　(2) BFS算法





算法是否正确？算法可否用来判定某图的连通性？



The complete list of all trees on 2,3,4 labeled vertices: $2^{2-2} = 1$ tree with 2 vertices, $3^{3-2} = 3$ trees with 3 vertices and $4^{4-2} = 16$ trees with 4 vertices.

What is the number T_n of different trees that can be formed from a set of n distinct vertices?

Cayley's formula gives the answer $T_n = n^{n-2}$.

Cayley定理： n个顶点的标号完全图K_n有n^{n-2}棵生成树

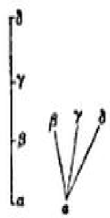
The Collected Mathematical Pappers of Arthur Cayley

895.

A THEOREM ON TREES.

[From the *Quarterly Journal of Pure and Applied Mathematics*, vol. XXIII. (1889), pp. 376—378.]

THE number of trees which can be formed with $n+1$ given knots $\alpha, \beta, \gamma, \ldots$ is $=(n+1)^{n-1}$; for instance $n=3$, the number of trees with the 4 given knots $\alpha, \beta, \gamma, \delta$ is $4^2=16$, for in the first form shown in the figure the $\alpha, \beta, \gamma, \delta$ may be arranged



$12+4=16$

in 12 different orders ($\alpha\beta\gamma\delta$ being regarded as equivalent to $\delta\gamma\beta\alpha$), and in the second form any one of the 4 knots $\alpha, \beta, \gamma, \delta$ may be in the place occupied by the α: the whole number is thus $12+4, =16$.

Considering for greater clearness a larger value of n, say $n=5$, I state the particular case of the theorem as follows:

No. of trees $(\alpha, \beta, \gamma, \delta, \epsilon, \zeta)$ = No. of terms of $(\alpha+\beta+\gamma+\delta+\epsilon+\zeta)^4\alpha\beta\gamma\delta\epsilon\zeta, =6^4, =1296$,

and it will be at once seen that the proof given for this particular case is applicable for any value whatever of n.

I use for any tree whatever the following notation: for instance, in the first of the forms shown in the figure, the branches are $\alpha\beta$, $\beta\gamma$, $\gamma\delta$; and the tree is said to be $\alpha\beta^2\gamma^2\delta$ (viz. the knots α, δ occur each once, but β, γ each twice); similarly in the second of the same forms, the branches are $\alpha\beta$, $\alpha\gamma$, $\alpha\delta$, and the tree is said

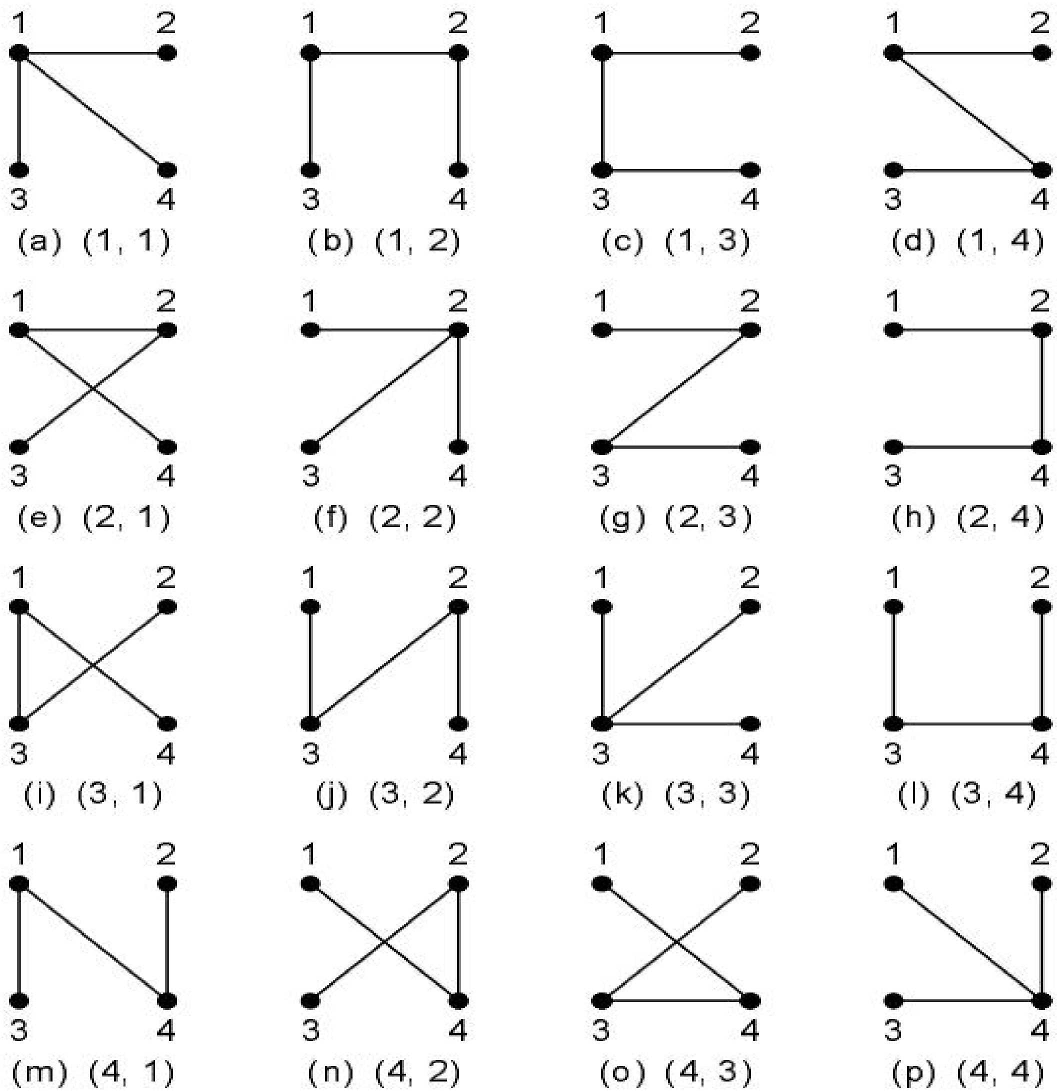
Prüfer sequence

From Wikipedia, the free encyclopedia

In combinatorial mathematics, the **Prüfer sequence** (also **Prüfer code** or **Prüfer numbers**) of a labeled tree is a unique sequence associated with the tree. The sequence for a tree on n vertices has length $n-2$, and can be generated by a simple iterative algorithm. Prüfer sequences were first used by Heinz Prüfer to prove Cayley's formula in 1918. [1]

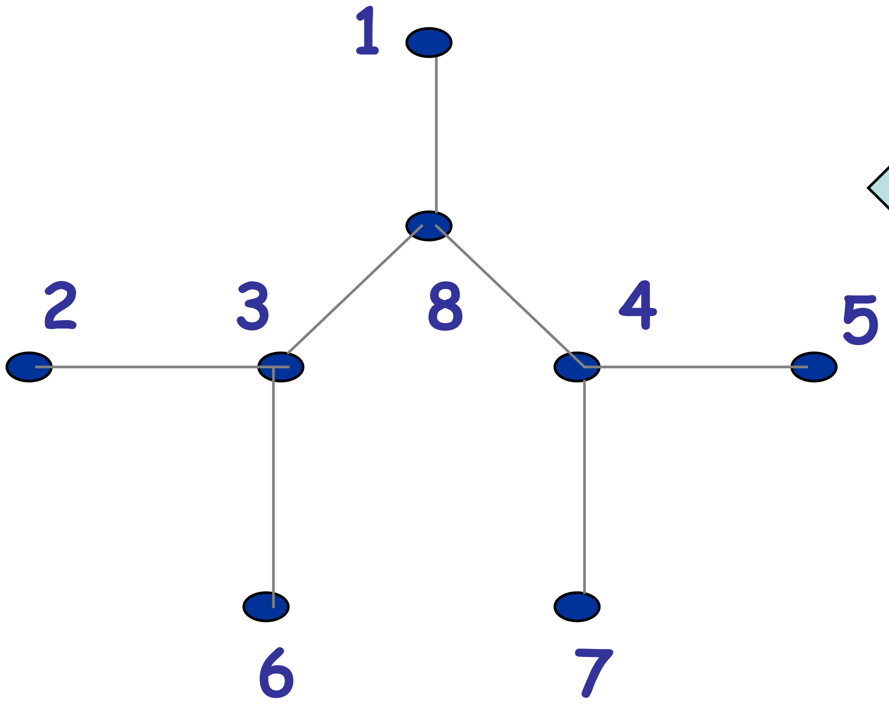
2 生成树 Spanning Tree

Example



All 16 spanning trees of K_4.

方法1 Prüfer code



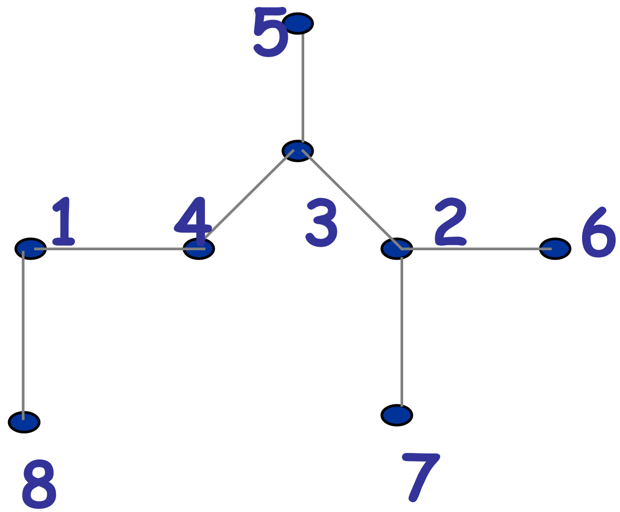
Prüfer Code

(8, 3, 4, 3, 8, 4)

↕ ↕ ↕ ↕ ↕ ↕

(1, 2, 5, 6, 3, 7)

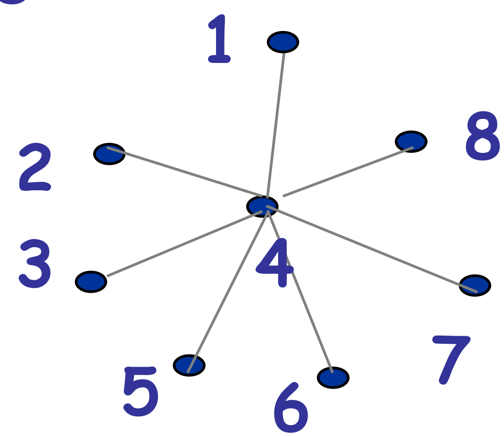
$$f: T_n \longrightarrow P_n, \quad f^{-1}: P_n \longrightarrow T_n$$



$\Longleftrightarrow$ (3, 2, 2, 3, 4, 1)

$\updownarrow$

(5, 6, 7, 2, 3, 4)



$\Longleftrightarrow$ (4, 4, 4, 4, 4, 4)

$\updownarrow$

(1, 2, 3, 5, 6, 7)

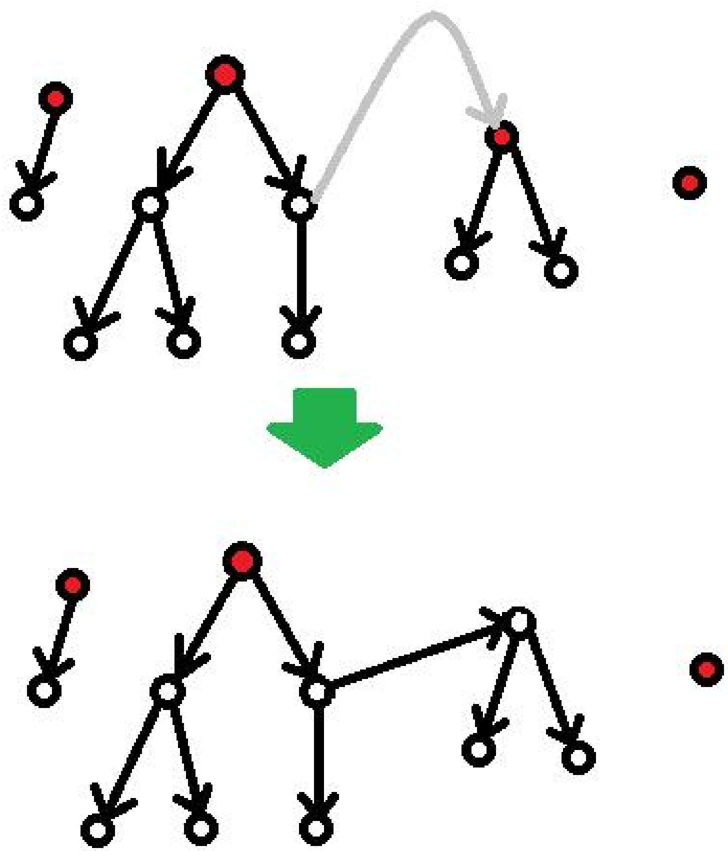
方法2 Double Counting Proof

Pitman's proof counts in two different ways the number of different sequences of directed edges that can be added to an empty graph on n vertices to form a rooted tree.

First way: $T_n \cdot n \cdot (n-1)!$

Second way: $\prod_{k=1}^{n-1} n_k = n^{n-1}(n-1)!$

At every step k = 1, . . . , n-1, let n_k be the number of possible directed edges from which to choose the edge to add: $n_k = n(n-k)$.



Adding a directed edge to a rooted forest with 4 trees.

方法3 Matrix-Tree Theorem

Let G be a graph with an orientation and A = A(G). The Laplacian matrix L(G) = L is the n × n matrix defined by

$$L_{ij} = \begin{cases} -A_{ij} & \text{if } i \neq j \\ \deg(v_i) & \text{if } i = j \end{cases}$$

Let K_n be a complete graph.

$$A = \begin{bmatrix} 0 & & & 1 \\ & 0 & & \\ & & \ddots & \\ & & 0 & \\ 1 & & & 0 \end{bmatrix} \qquad L = \begin{bmatrix} n-1 & & & & -1 \\ & n-1 & & & \\ & & \ddots & & \\ & & & n-1 & \\ -1 & & & & n-1 \end{bmatrix}$$

Equivalently,
$L = D - A$,
where D is a diagonal matrix with $D_{jj} = \deg(v_j)$ and A is the graph's adjacency matrix.

Observation.

The Laplacian matrix of a graph carries the same information as the adjacency matrix obviously, but has different useful and important properties, many relating to its spectrum.

(a) We have $MM^t = L$, where M denotes the incidence matrix.

(b) In M, each column entries sum to zero, so rows of M sum to zero vector, hence rank(M) < n. In L, entries in each column or row sum to zero, and L is symmetric.

(c) If G is regular of degree d (i.e., every vertex of G has degree d), then $L(G) = D - A(G)$, where A(G) denotes the adjacency matrix of G. Hence if G (or A(G)) has

We call $\lambda_1, \lambda_2, \ldots, \lambda_n$ the spectrum of graph G.

eigenvalues $\lambda_1, \ldots, \lambda_n$, then L(G) has eigenvalues $d - \lambda_1, \ldots, d - \lambda_n$.

(d) $\boxed{\mathrm{spec}(K_n) = [\underbrace{-1, \ldots, -1}_{n-1}, n-1].}$ $\boxed{\mathrm{specL}(K_n) = [0, \underbrace{n, \ldots, n}_{n-1}].}$

(Kirchoff's Matrix-Tree Theorem, 1847)
Let G be (finite connected) graph (without loops), and let L = L(G). Denote by L_0 the matrix obtained by removing the last row and column of L. Then

$$\boxed{\det(L_0) = \kappa(G) \qquad \text{Cauchy-Binet theorem}}$$

Corollary 1 Let G be a connected (loopless) graph with p vertices. Suppose that the eigenvalues of L(G) are $\lambda_1, \ldots, \lambda_{p-1}, \lambda_p$, with $\lambda_p = 0$. Then

$$\kappa(G) = \frac{1}{p}\,\lambda_1\lambda_2 \ldots\ \lambda_{p-1}\,.$$

Corollary 2 Suppose that G is also regular of degree d, and that the eigenvalues of A(G) are $\lambda_1, \ldots, \lambda_{p-1}, \lambda_p$, with $\lambda_p = d$. Then

$$\kappa(G) = \frac{1}{p}(d-\lambda_1)(d-\lambda_2)\cdots(d-\lambda_{p-1}).$$

Theorem. $\kappa(K_n) = n^{n-2}$

$\mathrm{spec}(K_n) = [\underbrace{-1, \ldots, -1}_{n-1}, n-1].$

Proof. K_n is regular of degree d, and $A(K_n)$ has eigenvalues -1($n-1$ times) and $n-1$(once). So from our Corollary we have $\kappa(K_n) = (1/n)((n-1) - (-1))^{n-1} = n^{n-2}$, as desired.

$\mathrm{specL}(K_n) = [0, \underbrace{n, \ldots, n}_{n-1}].$

This calculation(Matrix-Tree Theorem) was first devised by Gustav Kirchoff in 1847 as a way of obtaining values of current flow in electrical networks.

(Matrices were first emerging as a powerful mathematical tool about the same time.)

Laplacian Matrix——>Spectral Graph Theory

Other applications:

Nonlinear dimensionality reduction(Laplacian Eigenmaps)

Spectral clustering(graph partitionning)

Data classification

……