《算法复杂性理论》
第5讲 典型问题与状态空间树

山东师范大学信息科学与工程学院
段会川
2015年10月

目录

0-1背包问题—描述

0-1背包问题可描述为：n个物品和1个背包。对物品i，其价值为v_i，重量为w_i，背包的容量为W。如何选取物品装入背包，使背包中所装入的物品的总价值最大？

该问题为何被称为0-1背包问题呢？因为，在选择装入背包的物品时，对于物品i只有两种选择，即装入背包或不装入背包。不能将物品i装入背包多次，也不能只装入物品i的一部分。假设x_i表示物品i被装入背包的情况，当$x_i=0$时，表示物品没有被装入背包；当$x_i=1$时，表示物品被装入背包。

根据问题描述，设计出如下的约束条件和目标函数。

约束条件：$\begin{cases}\sum_{i=1}^{n} w_i x_i \leqslant W \\ x_i \in \{0,1\} \quad 1 \leqslant i \leqslant n\end{cases}$

目标函数：$\max \sum_{i=1}^{n} v_i x_i$

于是，问题归结为寻找一个满足约束条件(4-7)，并使目标函数(4-8)达到最大的解向量$X=(x_1,x_2,\cdots,x_n)$。

现实生活中，该问题可被表述成许多工业场合的应用，如资本预算、货物装载和存储分配等问题，因此对该问题的研究具有很重要的现实意义和实际价值。

0-1背包问题—示例

编号	1	2	3	4	5
重量	2	2	6	5	4
价值	6	3	5	4	6

背包容量：10 王秋芬，P101

解：1,1,0,0,1
重量：2+2+4=8
价值：6+3+6=15

背包问题的一个例子：应该选择哪些盒子，才能使价格尽可能地大，而保持重量小于或等于15 kg？

i	1	2	3	4
w_i	5	4	6	3
v_i	10	40	30	50

$W=10$
$X^*=(0,1,0,1)$
$W^*=4+3=7$
$V^*=40+50=90$

http://www.es.ele.tue.nl/education/5MC10/Solutions/knapsack.pdf

0-1背包问题—形式化定义

- 给定n个重量为$w_1,w_2,\cdots,w_n$价值为$v_1,v_2,\cdots,v_n$的物品和容量为W的背包，其中$W<\sum_{i=1}^{n} w_i$且物品不可分割，问怎样装入物品可以获得最大的价值？
- 以$x_1,x_2,\cdots,x_n$表示物品的装入情况，其中$x_i\in\{0,1\}$，则0-1背包问题可以表达为如下所示的优化问题：

$$\max_{x_1,x_2,\cdots,x_n} V(x_1,x_2,\cdots,x_n)=\sum_{i=1}^{n} x_i v_i,$$

$$s.t. \quad \sum_{i=1}^{n} x_i w_i \leq W,$$

$$x_i\in\{0,1\}, i=1,2,\cdots,n.$$

0-1背包问题—NP难的

- 背包问题是计算机科学中的一个重要问题：
  - 其判定性问题，即：
    总重不超过W的价值至少为V的解是否存在？
    是一个NP完全问题，即不可能存在所有实例上都能既正确又快速的多项式时间算法，除非P=NP.
  - 而其最优化问题是NP难的，它的求解至少与判定性问题一样难，即目前尚未找到判定给定的解是否为最优解的多项式时间算法。
  - 存在一个动态规划的伪多项式时间算法。
  - 存在一个基于伪多项式时间算法的完全的多项式近似算法。

TSP问题—描述

（1）问题描述。

设有 n 个城市组成的交通图，一个售货员从住地城市出发，到其他城市各一次去推销货物，最后回到住地城市。假定任意两个城市 i,j 之间的距离 d_{ij}（$d_{ij}=d_{ji}$）是已知的，问应该怎样选择一条最短的路线？

P152, 例5-10

（2）问题分析。

旅行商问题给定 n 个城市组成的无向带权图 $G=(V,E)$，顶点代表城市，权值代表城市之间的路径长度。要求找出以住地城市开始的一个排列，按照这个排列的顺序推销货物，所经路径长度是最短的。

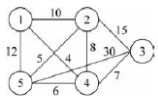
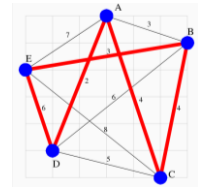


图 5-35　无向带权图

P153

TSP问题—示例



	A	B	C	D	E
A	0	3	4	2	7
B		0	4	6	3
C			0	5	8
D				0	6
E					0

http://people.sc.fsu.edu/~jburkardt/latex/genetic_2013_fsu/genetic_2013_fsu.html

TSP问题—示例

【例 5-1】　给定一个有向带权图 $G=(V,E)$，权非负，如图 5-1 所示。找出顶点 1→5 的最短路径及其长度。

问题分析：该问题所有可能的路径只有三条，分别是 1→2→4→5，1→3→4→5，1→4→5，采用穷举搜索的方法逐一检查这三条路径的长度，得出的最短路径为 1→2→4→5，其长度为 3。

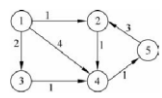
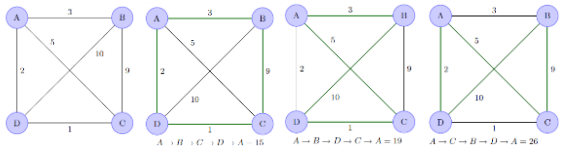


图 5-1　有向带权图



Integer linear programming formulation

For i = 0, ..., n, let u_i be an artificial variable, and finally take c_{ij} to be the distance from city i to city j. Then TSP can be written as the following integer linear programming problem:

$$\min \sum_{i=0}^{n} \sum_{j\neq i,j=0}^{n} c_{ij}x_{ij}$$

$$0 \le x_{ij} \le 1 \qquad i,j = 0,\cdots,n$$

$$u_i \in \mathbf{Z} \qquad i = 0,\cdots,n$$

$$\sum_{i=0,i\neq j}^{n} x_{ij} = 1 \qquad j = 0,\cdots,n$$

$$\sum_{j=0,j\neq i}^{n} x_{ij} = 1 \qquad i = 0,\cdots,n$$

$$u_i - u_j + nx_{ij} \le n-1 \quad 1 \le i \ne j \le n$$

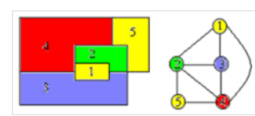
TSP问题—NP难

- TSP问题的判定性问题是一个NP完全问题
  - 给定n个城市及其间的距离，判定是否存在一条经过每个城市一次且距离小于L的路径
- TSP优化问题是一个NP难的问题
  - 穷举法：$O(n*n!)$
  - 动态规划法：$O(n^2 2^n)$

图着色问题

- 图着色问题（Graph Coloring Problem, GCP）　又称着色问题，是最著名的NP-完全问题之一。
- 数学定义：给定一个无向图G=（V, E），其中V为顶点集合，E为边集合，图着色问题即为将V分为K个颜色组，每个组形成一个独立集，即其中没有相邻的顶点。其优化版本是希望获得最小的K值。

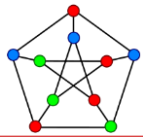
百度百科



图着色问题

- 图的m-着色是一个判定性问题，它是NP完全的
  - 给定无向连通图G和m种不同的颜色。用这些颜色为图G的各顶点着色，每个顶点着一种颜色，是否有一种着色法使G中任意相邻的2个顶点着不同颜色?
- 图的着色优化问题是NP难的
  - 求一个图的最小着色数m的问题称为图着色问题的优化问题。

百度百科



目录

状态空间树

无论是货郎担问题、还是背包问题，都有这样一个共同的特点，即所求解的问题都有 n 个输入，都能用一个 n 元组 $X=(x_1,x_2,\cdots,x_n)$ 来表示问题的解。其中，x_i 的取值范围为某个有穷集 S。例如，在 0/1 背包问题中，$S=\{0,1\}$；而在货郎担问题中，$S=\{1,2,\cdots,n\}$。一般，把 $X=(x_1,x_2,\cdots,x_n)$ 称为问题的解向量；而把 x_i 的所有可能取值范围的组合，称为问题的解空间。例如，当 $n=3$ 时，0/1 背包问题的解空间是：

$$\{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1)\}$$

它有 8 种可能的解。当输入规模为 n 时，它有 2^n 种可能的解。而在当 $n=3$ 时的货郎担问题中，x_i 的取值范围 $S=\{1,2,3\}$。于是，在这种情况下，货郎担问题的解空间是：

$$\{(1,1,1),(1,1,2),(1,1,3),(1,2,1),(1,2,2),(1,2,3),\ \cdots,(3,3,1),(3,3,2),(3,3,3)\}$$

它有 27 种可能的解。当输入规模为 n 时，它有 n^n 种可能的解。考虑到货郎担问题的解向量 $X=(x_1,x_2,\cdots,x_n)$ 中，必须满足约束方程 $x_i\neq x_j$，因此可以把货郎担问题的解空间压缩为：

$$\{(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2),(3,2,1)\}$$

它有 6 种可能的解。当输入规模为 n 时，它有 $n!$ 种可能的解。

0-1背包问题的状态空间树—子集树

子集树是使用回溯法解题时经常遇到的一种典型的解空间树。当所给的问题是从 n 个元素组成的集合 S 中找出满足某种性质的一个子集时，相应的解空间树称为子集树。此类问题解的形式为 n 元组 $(x_1,x_2,\cdots,x_n)$，分量 $x_i(i=1,2,\cdots,n)$ 表示第 i 个元素是否在要找的子集中。x_i 的取值为 0 或 1，$x_i=0$ 表示第 i 个元素不在要找的子集中；$x_i=1$ 表示第 i 个元素在要找的子集中。如图 5-15 所示是 $n=3$ 时的子集树。

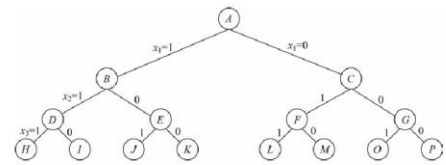


图 5-15　$n=3$ 时的子集树

P133

0-1背包问题的状态空间树—子集树

子集树中所有非叶子结点均有左右两个分支，左分支为 1，右分支为 0，反之也可以。本书约定子集树的左分支为 1，右分支为 0。树中从根到叶子的路径描述了一个 n 元 0-1 向量，这个 n 元 0-1 向量表示集合 S 的一个子集，这个子集由对应分量为 1 的元素组成。如假定 3 个元素组成的集合 S 为 $\{1,2,3\}$，从根结点 A 到叶结点 I 的路径描述的 n 元组为 $(1,1,0)$，它表示 S 的一个子集 $\{1,2\}$。从根结点 A 到叶结点 M 的路径描述的 n 元组为 $(0,1,0)$，它表示 S 的另一个子集 $\{2\}$。

在子集树中，树的根结点表示初始状态，中间结点表示某种情况下的中间状态，叶子结点表示结束状态。分支表示从一个状态过渡到另一个状态的行为。从根结点到叶子结点的路径表示一个可能的解。子集树的深度等于问题的规模。

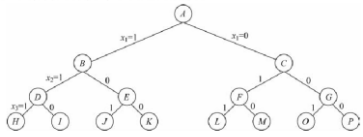


图 5-15　$n=3$ 时的子集树

P133

0-1背包问题的状态空间树—子集树

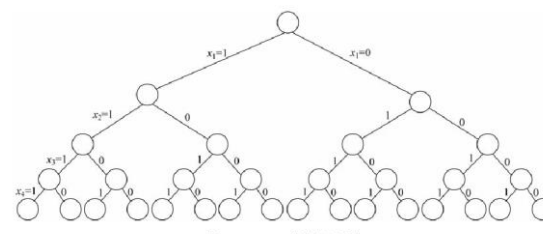


图 5-16　$n=4$ 时的解空间树

P136

解空间为子集树的问题

0-1 背包问题：从 n 个物品组成的集合 S 中找出一个子集，这个子集内所有物品的总重量不超过背包的容量，并且这些物品的总价值在 S 的所有不超过背包容量的子集中是最大的。显然，这个问题的解空间树是一棵子集树。

子集和问题：给定 n 个整数和一个整数 C，要求找出 n 个数中哪些数相加的和等于 C。这个问题实质上是要求从 n 个数组成的集合 S 中找出一个子集，这个子集中所有数的和等于给定的 C。因此，子集和问题的解空间树也是一棵子集树。

装载问题：n 个集装箱要装上两艘载重量分别为 c_1 和 c_2 的轮船，其中集装箱 i 的重量为 w_i，且 $\sum_{i=1}^{n} w_i \leqslant c_1 + c_2$。装载问题要求确定是否有一个合理的装载方案可将这个集装箱装上这两艘轮船，如果有，找出一种装载方案。

P134

Subset sum problem

- **In computer science, the subset sum problem is one of the important problems in complexity theory and cryptography. The problem is this: given a set (or multiset) of integers, is there a non-empty subset whose sum is zero? For example, given the set {−7, −3, −2, 5, 8}, the answer is yes because the subset {−3, −2, 5} sums to zero. The problem is NP-complete.**

Subset sum problem

- **An equivalent problem is this: given a set of integers and an integer s, does any non-empty subset sum to s? Subset sum can also be thought of as a special case of the knapsack problem.[1] One interesting special case of subset sum is the partition problem, in which s is half of the sum of all elements in the set.**

解空间为子集树的问题

最大团问题：给定一个无向图，找出它的最大团。这个问题等价于从给定无向图的 n 个顶点组成的集合中找出一个顶点子集，这个子集中的任意两个顶点之间有边相连且包含的顶点个数是所有该类子集中包含顶点个数最多的。因此这个问题也是从整体中取出一部分，这一部分构成整体的一个子集且满足一定的特性，它的解空间树是一棵子集树。

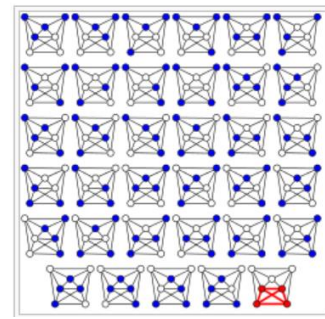
可见，对于要求从整体中取出一部分，这一部分需要满足一定的特性，整体与部分之间构成包含与被包含的关系，即子集关系的一类问题，均可采用子集树来描述它们的解空间树。这类问题在解题时可采用统一的算法设计模式。

P134

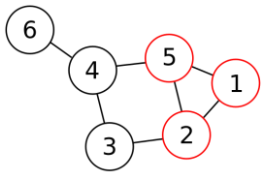
Clique problem

- **In computer science, the clique problem refers to any of the problems related to finding particular complete subgraphs ("cliques") in a graph, i.e., sets of elements where each pair of elements is connected.**
- **For example, the maximum clique problem arises in the following real-world setting. Consider a social network, where the graph's vertices represent people, and the graph's edges represent mutual acquaintance. To find a largest subset of people who all know each other, one can systematically inspect all subsets, a process that is too time-consuming to be practical for social networks comprising more than a few dozen people.**

Clique problem



Clique problem



The graph shown has one maximum clique, the triangle {1,2,5}, and four more maximal cliques, the pairs {2,3}, {3,4}, {4,5}, and {4,6}

Clique problem

- **Although this brute-force search can be improved by more efficient algorithms, all of these algorithms take exponential time to solve the problem. Therefore, much of the theory about the clique problem is devoted to identifying special types of graph that admit more efficient algorithms, or to establishing the computational difficulty of the general problem in various models of computation.[1] Along with its applications in social networks, the clique problem also has many applications in bioinformatics and computational chemistry**

Clique problem

- **Clique problems include:**
  - **finding a maximum clique (largest clique by vertices),**
  - **finding a maximum weight clique in a weighted graph,**
  - **listing all maximal cliques (cliques that cannot be enlarged)**
  - **solving the decision problem of testing whether a graph contains a clique larger than a given size.**
- **These problems are all hard: the clique decision problem is NP-complete (one of Karp's 21 NP-complete problems),**

TSP问题的状态空间树—排列树

排列树是用回溯法解题时经常遇到的第二种典型的解空间树。当所给的问题是从 n 个元素的排列中找出满足某种性质的一个排列时，相应的解空间树称为排列树。此类问题解的形式为 n 元组 $(x_1,x_2,\cdots,x_n)$，分量 $x_i(i=1,2,\cdots,n)$ 表示第 i 个位置的元素是 x_i。n 个元素组成的集合为 $S=\{1,2,\cdots,n\}$，$x_i\in S-\{x_1,x_2,\cdots,x_{i-1}\}$，$i=1,2,\cdots,n$。

$n=3$ 时的排列树如图 5-26 所示：

在排列树中从根到叶子的路径描述了 n 个元素的一个排列。如 3 个元素的位置为 $\{1,2,3\}$，从根结点 A 到叶结点 L 的路径描述的一个排列为 $(1,3,2)$，即第 1 个位置的元素是 1，第 2 个位置的元素是 3，第 3 个位置的元素是 2；从根结点 A 到叶结点 M 的路径描述的一个排列为 $(2,1,3)$；从根结点 A 到叶结点 P 的路径描述的一个排列为 $(3,2,1)$。

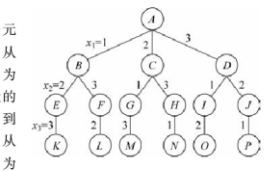


图 5-26 $n=3$ 的排列树

P146

TSP问题的状态空间树—排列树

在排列树中，树的根结点表示初始状态（所有位置全部没有放置元素）；中间结点表示某种情况下的中间状态（中间结点之前的位置上已经确定了元素，中间结点之后的位置上还没有确定元素）；叶子结点表示结束状态（所有位置上的元素全部确定）；分支表示从一个状态过渡到另一个状态的行为（在特定位置上放置元素）；从根结点到叶子结点的路径表示一个可能的解（所有元素的一个排列）。排列树的深度等于问题的规模。

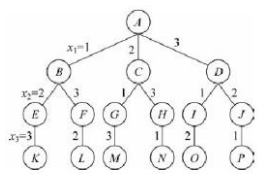


图 5-26 $n=3$ 的排列树

P146

TSP问题的状态空间树—排列树

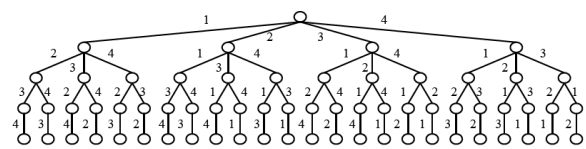


图 7.1 n=4 时货郎担问题的状态空间树

解空间为排列树的问题

n 皇后问题：满足显约束为不同行、不同列的解空间树。约定不同行的前提下，n 个皇后的列位置是 n 个列的一个排列，这个排列必须满足 n 个皇后的位置不在一条斜线上。

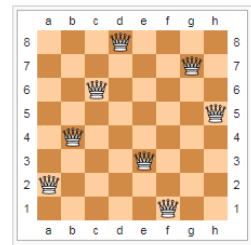
旅行商问题：找出 n 个城市的一个排列，沿着这个排列的顺序遍历 n 个城市，最后回到出发城市，求长度最短的旅行路径。

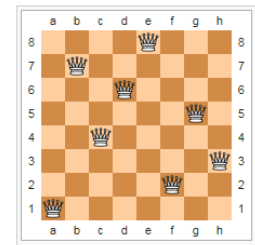
批处理作业调度问题：给定 n 个作业的集合$\{J_1,J_2,\cdots,J_n\}$，要求找出 n 个作业的一个排列，按照这个排列进行调度，使得完成时间和达到最小。

圆排列问题：给定 n 个大小不等的圆 $c_1,c_2,\cdots,c_n$，现要将这 n 个圆放入一个矩形框中，且要求各圆与矩形框的底边相切。圆排列问题要求从 n 个圆的所有排列中找出具有最小长度的圆排列。

P147

8-皇后问题





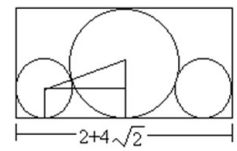
批处理作业调度问题

给定 n 个作业的集合 J = { J1，J2，…，Jn }。每一个作业 Ji 都有两项任务分别在两台机器上完成。每个作业必须先由机器 1 处理，然后由机器 2 处理。作业 Ji 需要机器 j 的处理时间为 tji，其实 i=1，2，…，n，j=1，2。对于一个确定的作业调度，设 Fji 是作业 i 在机器 j 上完成处理的时间。所有作业在机器 2 上完成处理的时间和称为该作业调度的完成时间和。

批处理作业调度问题要求对于给定的 n 个作业，制定最佳作业调度方案，使其完成时间和达到最小。

圆排列问题

给定n个大小不等的圆c1,c2,…,cn，现要将这n个圆排进一个矩形框中，且要求各圆与矩形框的底边相切。圆排列问题要求从n个圆的所有排列中找出有最小长度的圆排列。例如，当n=3，且所给的3个圆的半径分别为1，1，2时，这3个圆的最小长度的圆排列如图所示。其最小长度为 $2+4\sqrt{2}$ 。



解空间为排列树的问题

电路板排列问题：将 n 块电路板以最佳排列方式插入带有 n 个插槽的机箱中。n 块电路板的不同排列方式对应于不同的电路板插入方案。设 $B=\{1,2,\cdots,n\}$ 是 n 块电路板的集合，$L=\{N_1,N_2,\cdots,N_m\}$ 是连接这 n 块电路板中若干电路板的 m 个连接块。N_i 是 B 的一个子集，且 N_i 中的电路板用同一条导线连接在一起。设 x 表示 n 块电路板的一个排列，即在机箱的第 i 个插槽中插入的电路板编号是 x_i。x 所确定的电路板排列 $\mathrm{Density}(x)$ 密度定义为：跨越相邻电路板插槽的最大连线数。在设计机箱时，插槽一侧的布线间隙由电路板排列的密度确定。因此，电路板排列问题要求对于给定的电路板连接条件，确定电路板的最佳排列，使其具有最小排列密度。

可见，对于要求从 n 个元素中找出它们的一个排列，该排列需要满足一定的特性这类问题，均可采用排列树来描述它们的解空间结构。这类问题在解题时可采用统一的算法设计模式。

P147

图着色问题的状态空间树—满m叉树

满 m 叉树是用回溯法解题时经常遇到的第三种典型的解空间树，也可以称为组合树。当所给问题的 n 个元素中每一个元素均有 m 种选择，要求确定其中的一种选择，使得对这 n 个元素的选择结果组成的向量满足某种性质，即寻找满足某种特性的 n 个元素取值的一种组合。这类问题的解空间树称为满 m 叉树。此类问题解的形式为 n 元组$(x_1,x_2,\cdots,x_n)$，分量 $x_i(i=1,2,\cdots,n)$ 表示第 i 个元素的选择为 x_i。$n=3$ 时的满 $m=3$ 叉树如图 5-41 所示。

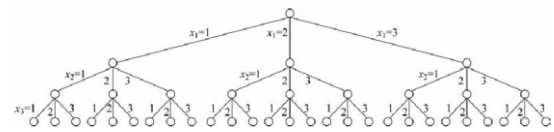


图 5-41　满 3 叉树

P157

图着色问题的状态空间树—满m叉树

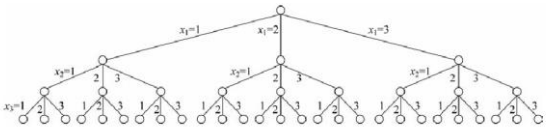


图 5-41　满 3 叉树

在满 m 叉树中从根到叶子的路径描述了 n 个元素的一种选择。树的根结点表示初始状态(任何一个元素都没有确定),中间结点表示某种情况下的中间状态(一些元素的选择已经确定,另一些元素的选择没有确定),叶子结点表示结束状态(所有元素的选择均已确定),分支表示从一个状态过渡到另一个状态的行为(特定元素做何种选择),从根结点到叶子结点的路径表示一个可能的解(所有元素的一个排列)。满 m 叉树的深度等于问题的规模 n。

P157

状态空间树为满m叉树的问题

n 皇后问题：显约束为不同行的解空间树,在不同行的前提下,任何一个皇后的列位置都有 n 种选择。n 个列位置的一个组合必须满足 n 个皇后的位置不在同一列或不在同一条斜线上的性质。这个问题的解空间便是一棵满 $m(m=n)$ 叉树。

图的 m 着色问题：给定无向连通图 G 和 m 种不同的颜色。用这些颜色为图 G 的各顶点着色,每个顶点着一种颜色。如果有一种着色法使 G 中有边相连的两个顶点着不同颜色,则称这个图是 m 可着色的。图的 m 着色问题是对于给定图 G 和 m 种颜色,找出所有不同的着色法。这个问题实质上是用给定的 m 种颜色给无向连通图 G 的顶点着色。每一个顶点所着的颜色有 m 种选择,找出 n 个顶点着色的一个组合,使其满足有边相连的两个顶点之间所着颜色不相同。很明显,这是一棵满 m 叉树。

P157

状态空间树为满m叉树的问题

最小重量机器设计问题可以看做给机器的 n 个部件找供应商,也可以看做 m 个供应商供应机器的哪个部件。如果看做给机器的 n 个部件找供应商,则问题实质为：n 个部件中的每一个部件均有 m 种选择,找出 n 个部件供应商的一个组合,使其满足 n 个部件的总价格不超过 c 且总重量是最小的。问题的解空间是一棵满 m 叉树。如果看做 m 个供应商供应机器的哪个部件,则问题的解空间是一棵排列树,读者可以自己思考一下原因。

可见,对于要求找出 n 个元素的一个组合,该组合需要满足一定特性这类问题,均可采用满 m 叉树来描述它们的解空间结构。这类问题在解题时可采用统一的算法设计模式。

P158

目录

The End

Thanks !