



## 第5章 回溯与分支限界

5.1 回溯算法的基本思想和适用条件

5.2 回溯算法的设计步骤

5.3 回溯算法的效率估计和改进途径

5.4 分支限界

5.4.1 背包问题

5.4.2 最大团问题

5.4.3 货郎问题

5.4.4 圆排列问题

5.4.5 连续邮资问题



北京大学



## 5.1 回溯算法的基本思想和适用条件

### 5.1.1 几个典型例子

四后问题

0-1背包问题

货郎问题（TSP）

### 5.1.2 回溯算法的适用条件



北京大学

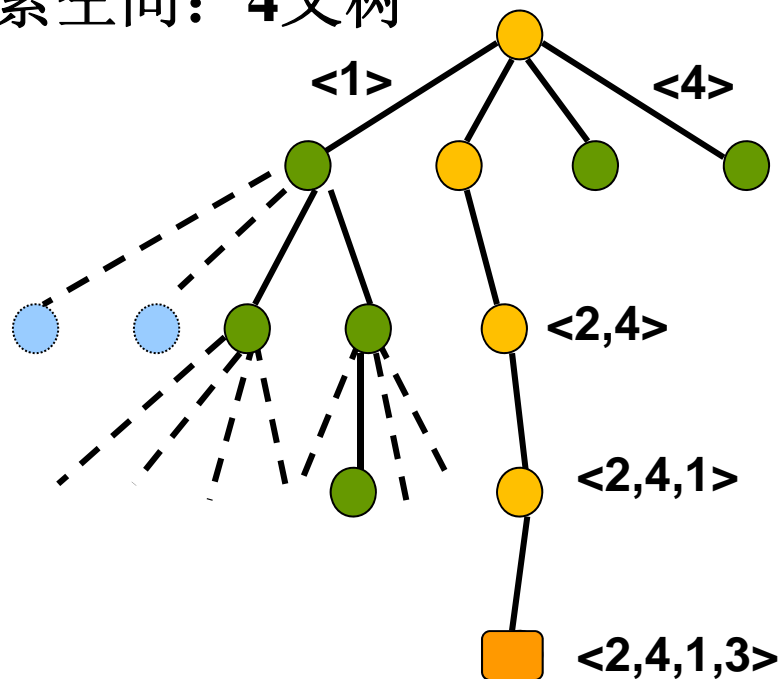


## 5.1.1几个典型例子

### 例1 $n$ 后问题

4后问题：解是一个4维向量， $\langle x_1, x_2, x_3, x_4 \rangle$ （放置列号）

搜索空间：4叉树



	○		
			○
○			
		○	

8后问题：解是一个8维向量， $\langle x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \rangle$

搜索空间：8叉树，一个解： $\langle 1, 3, 5, 2, 4, 6, 8, 7 \rangle$



北京大学



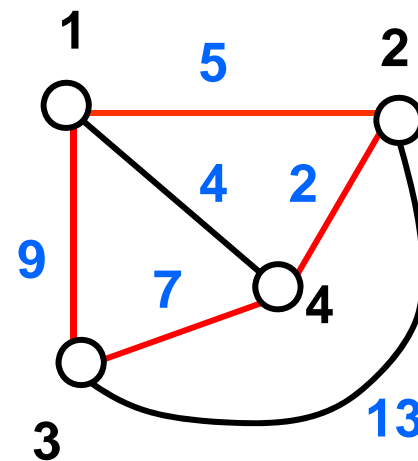
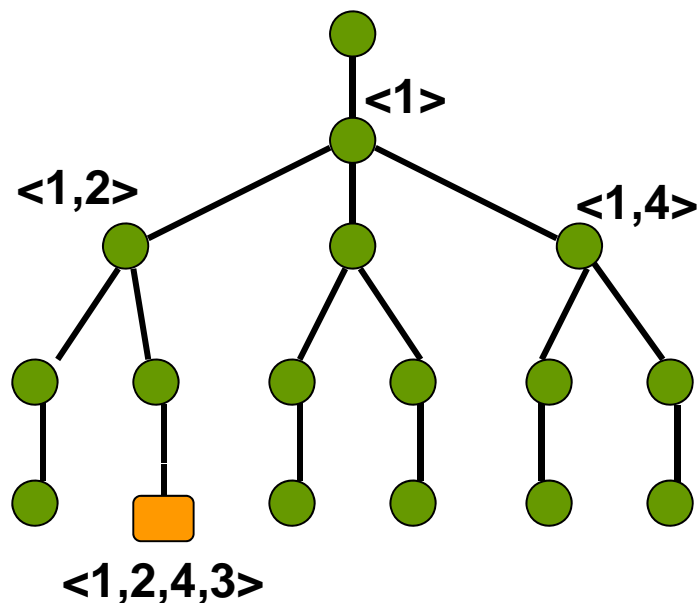


## 例3：货郎问题

$\langle i_1, i_2, \dots, i_n \rangle$  为巡回路线

搜索空间：排列树,  $(n-1)!$  片树叶

实例：



$\langle 1, 2, 4, 3 \rangle$  对应于巡回路线：1  $\rightarrow$  2  $\rightarrow$  4  $\rightarrow$  3  $\rightarrow$  1

长度：5+2+7+9=23



北京大学



# 回溯算法的基本思想

- (1) 适用问题：求解搜索问题和优化问题
- (2) 搜索空间：树，结点对应部分解向量，树叶对应可行解
- (3) 搜索过程：采用系统的方法隐含遍历搜索树
- (4) 搜索策略：深度优先，宽度优先，函数优先，宽深结合等
- (5) 结点分支判定条件：
  - 满足约束条件---分支扩张解向量
  - 不满足约束条件，回溯到该结点的父结点
- (6) 结点状态：动态生成
  - 白结点(尚未访问);
  - 灰结点(正在访问该结点为根的子树);
  - 黑结点(该结点为根的子树遍历完成)
- (7) 存储：当前路径







## 5.1.2 回溯算法的适用条件

设  $P(x_1, x_2, \dots, x_i)$  为真表示向量  $\langle x_1, x_2, \dots, x_i \rangle$  满足某个性质 ( $n$ 后问题中 $i$ 个皇后放置在彼此不能攻击的位置)

多米诺性质:

$$P(x_1, x_2, \dots, x_{k+1}) \rightarrow P(x_1, x_2, \dots, x_k) \quad 0 < k < n$$

例4 求不等式的整数解

$$5x_1 + 4x_2 - x_3 \leq 10, \quad 1 \leq x_i \leq 3, \quad i=1,2,3$$

$P(x_1, \dots, x_k)$ : 意味将  $x_1, x_2, \dots, x_k$  代入原不等式的相应部分  
使得左边小于等于10

不满足多米诺性质

变换: 令  $x_3 = 3 - x_3'$ ,

$$5x_1 + 4x_2 + x_3' \leq 13, \quad 1 \leq x_1, x_2 \leq 3, \quad 0 \leq x_3' \leq 2$$



北京大学



## 5.2 回溯算法的设计步骤

(1) 定义搜索问题的解向量和每个分量的取值范围

解向量为  $\langle x_1, x_2, \dots, x_n \rangle$

确定  $x_i$  的可能取值的集合为  $X_i, i = 1, 2, \dots, n$ .

(2) 当  $x_1, x_2, \dots, x_{k-1}$  确定以后计算  $x_k$  取值集合  $S_k, S_k \subseteq X_k$

(3) 确定结点儿子的排列规则

(4) 判断是否满足多米诺性质

(5) 搜索策略----深度优先、宽度优先等

(6) 确定每个结点分支约束条件

(7) 确定存储搜索路径的数据结构



北京大学





# 回溯算法的递归实现

算法 **ReBack( $k$ )**

1. if  $k > n$  then  $\langle x_1, x_2, \dots, x_n \rangle$  是解
2. else while  $S_k \neq \emptyset$  do
3.      $x_k \leftarrow S_k$  中最小值
4.      $S_k \leftarrow S_k - \{x_k\}$
5.     计算  $S_{k+1}$
6.     **ReBack( $k+1$ )**

算法 **ReBacktrack( $n$ )**

输入:  $n$

输出: 所有的解

1. for  $k \leftarrow 1$  to  $n$  计算  $X_k$
2. **ReBack(1)**



北京大学



# 回溯算法的迭代实现

## 迭代算法 Backtrack

输入:  $n$

输出: 所有的解

1. 对于  $i = 1, 2, \dots, n$  确定  $X_i$
2.  $k \leftarrow 1$
3. 计算  $S_k$
4. while  $S_k \neq \emptyset$  do
5.      $x_k \leftarrow S_k$  中最小值;  $S_k \leftarrow S_k - \{x_k\}$
6.     if  $k < n$  then
7.          $k \leftarrow k + 1$ ; 计算  $S_k$
8.     else  $\langle x_1, x_2, \dots, x_n \rangle$  是解
9. if  $k > 1$  then  $k \leftarrow k - 1$ ; goto 4



北京大學



## 例5 装载问题

$n$ 个集装箱装上2艘载重分别为 $c_1$ 和 $c_2$ 的轮船， $w_i$ 为集装箱 $i$ 的重量，且

$$\sum_{i=1}^n w_i \leq c_1 + c_2$$

问是否存在一种合理的装载方案将 $n$ 个集装箱装上轮船？如果有，给出一种方案.

求解思路：令第一船装载量为 $W_1$ ，用回溯算法求使 $W_1 - c_1$ 达到最小的装载方案 $\langle x_1, x_2, \dots, x_n \rangle$ ，如果  $\sum_{i=1}^n w_i - W_1 \leq c_2$  回答Yes, 否则回答No.

问题满足多米诺性质，搜索策略：深度优先



北京大学



# 算法

## 算法5.4 Loading ( $W, c_1$ ),

输入：集装箱重量  $W = \langle w_1, w_2, \dots, w_n \rangle$ ,  $c_1$  是第一条船的载重

输出：使船1装载量最大的方案  $\langle x_1, x_2, \dots, x_n \rangle$ , 其中  $x_i = 0, 1, \dots, n$

1.  $\text{Sort}(W)$ ;     //对  $w_1, w_2, \dots, w_n$  按照从大到小排序
2.  $B \leftarrow \infty$ ;  $best \leftarrow \infty$ ;  $i \leftarrow 1$ ;
3. while  $i \leq n$  do
4.     if 装入  $i$  后重量不超过  $c_1$
5.     then  $B \leftarrow B - w_i$ ;  $x[i] \leftarrow 1$ ;  $i \leftarrow i + 1$ ;
6.     else  $x[i] \leftarrow 0$ ;  $i \leftarrow i + 1$ ;
7. if  $B < best$  then 记录解;  $Best \leftarrow B$ ;
8. Backtrack( $i$ );
9. if  $i = 1$  then return 最优解
10. else goto 3.



北京大学



# 实例

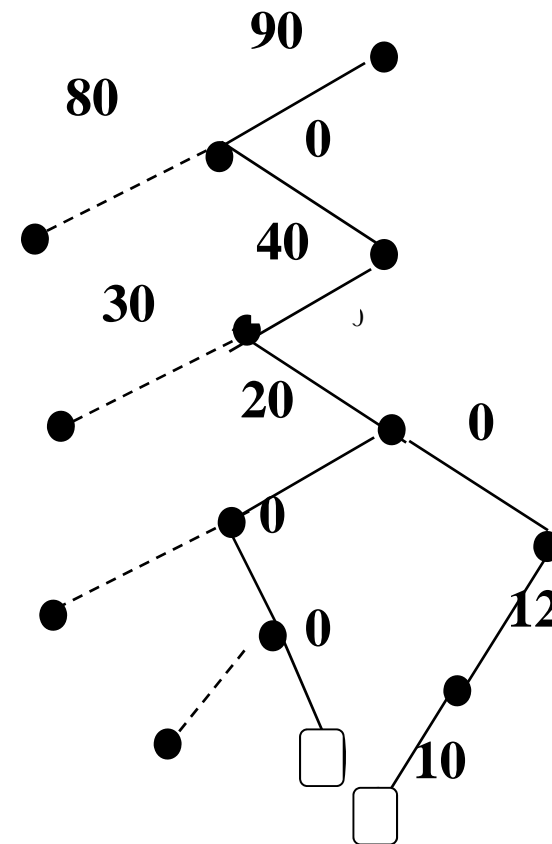
## 算法5.5 Backtrack( $i$ )

1. while  $i > 1$  and  $x[i] = 0$  do
2.      $i \leftarrow i - 1$ ;
3. if  $x[i] = 1$
4. then  $x[i] \leftarrow 0$ ;  $B \leftarrow B + w_i$ ;  $i \leftarrow i + 1$ .

$W = \langle 90, 80, 40, 30, 20, 12, 10 \rangle$ ,

$c_1 = 152$ ,  $c_2 = 130$

复杂性:  $W(n) = O(2^n)$



北京大学



## 例6 图的着色问题

问题：给定无向连通图 $G$ 和 $m$ 种颜色，用这些颜色给图的顶点着色，每个顶点一种颜色. 要求是： $G$ 的每条边的两个顶点着不同颜色. 给出所有可能的着色方案；如果不存在着这样的方案，则回答“**No**”.

则搜索空间为深度 $n$ 的 $m$ 叉完全树. 将颜色编号为 $1, 2, \dots, m$ ,  
结点 $\langle x_1, x_2, \dots, x_k \rangle$ :  $x_1, x_2, \dots, x_k \in \{1, 2, \dots, m\}$ ,  $1 \leq k \leq n$ , 表示顶点1着颜色 $x_1$ , 顶点2着颜色 $x_2$ , ..., 顶点 $k$ 着颜色 $x_k$ .

约束条件：该顶点邻接表中的顶点与该顶点没有同色；

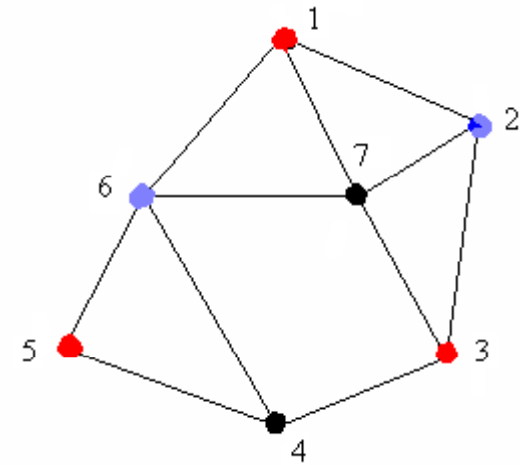
搜索策略：深度优先

时间： $O(nm^n)$



北京大学





$\langle 1, 2, 1, 3, 1, 2, 3 \rangle$



## 提高效率的途径

根据对称性，只需搜索 $1/3$ 的解空间即可. 当 $1$ 和 $2$ 确定, 即 $\langle 1, 2 \rangle$ 以后，只有 $1$ 个解，因此在 $\langle 1, 3 \rangle$ 为根的子树中也只有 $1$ 个解. 由于 $3$ 个子树的对称性，总共有 $6$ 个解.

进一步分析，在取定 $\langle 1, 2 \rangle$ 以后，不可以扩张成 $\langle 1, 2, 3 \rangle$ , 因为可以检查是否有和 $1, 2, 3$ 都相邻的顶点. 如果存在, 例如 $7$ , 则没有解. 所以可以从打叉的结点回溯，而不必搜索它的子树.





## 5.3 搜索树结点数估计

计数搜索树中的结点，Monte Carlo方法

### Monte Carlo方法

1. 从根开始，随机选择一条路径，直到不能分支为止，即从 $x_1, x_2, \dots$ ，依次对 $x_i$ 赋值，每个 $x_i$ 的值是从当时的 $S_i$ 中随机选取，直到向量不能扩张为止。
2. 假定搜索树的其他 $|S_i| - 1$ 个分支与以上随机选出的路径一样，计数搜索树的点数。
3. 重复步骤 1 和 2，将结点数进行概率平均。



北京大学



# 算法实现

## Monte Carlo

输入:  $n, t$  为正整数,  $n$  为皇后数,  $t$  为抽样次数

输出:  $sum$ , 即  $t$  次抽样路径长度的平均值

1.  $sum \leftarrow 0$       //  $sum$  为  $t$  次结点平均数
2. for  $i \leftarrow 1$  to  $t$  do    // 取样次数  $t$
3.     $m \leftarrow \text{Estimate}(n)$     //  $m$  为本次结点总数
4.     $sum \leftarrow sum + m$
5.  $sum \leftarrow sum / t$





# 子过程

$m$ 为输出——本次取样结点总数,  $k$  为层数,  $r_1$ 为本层分支数,  $r_2$ 为上层分支数,  $n$ 为树的层数

算法Estimate( $n$ )

1.  $m \leftarrow 1; r_2 \leftarrow 1; k \leftarrow 1$       //  $m$ 为结点总数
2. While  $k \leq n$  do
3.    if  $S_k = \emptyset$  then return  $m$
4.     $r_1 \leftarrow |S_k| * r_2$       //  $r_1$ 为扩张后结点总数
5.     $m \leftarrow m + r_1$       //  $r_2$ 为扩张前结点总数
6.     $x_k \leftarrow$  随机选择  $S_k$  的元素
7.     $r_2 \leftarrow r_1$
8.     $k \leftarrow k+1$





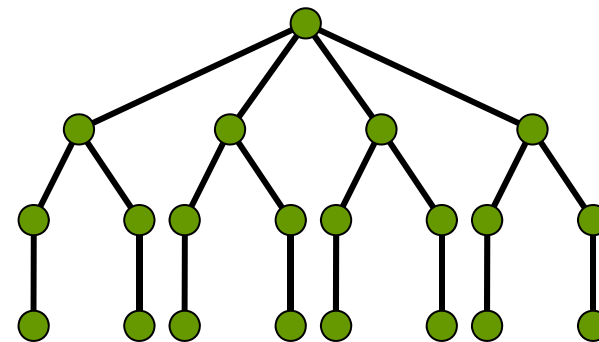
# 实例

估计四后搜索树的结点数

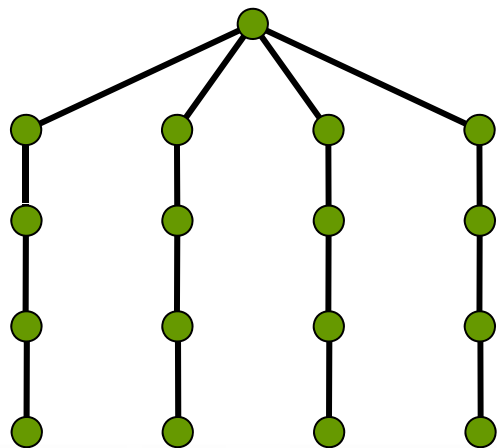
case1.  $\langle 1, 4, 2 \rangle$ :  $1 + 4 + 4 \times 2 + 4 \times 2 = 21$

case2.  $\langle 2, 4, 1, 3 \rangle$ :  $4 \times 4 + 1 = 17$

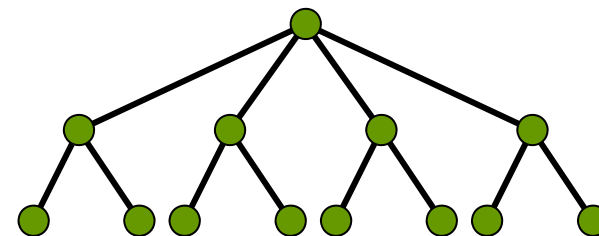
case3.  $\langle 1, 3 \rangle$ :  $1 + 4 \times 1 + 4 \times 2 = 13$



Case1:  $\langle 1, 4, 2 \rangle$



Case2:  $\langle 2, 4, 1, 3 \rangle$



Case3:  $\langle 1, 3 \rangle$



北京大学





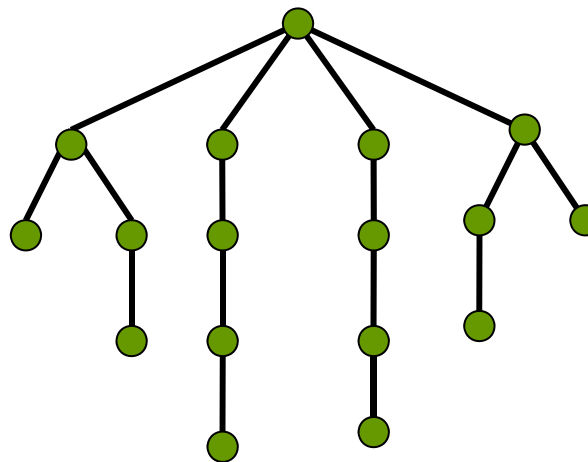
## 估计结果

假设 4 次抽样测试:

case1:1次, case2:1次, case3:2次,

平均结点数 =  $(21 \times 1 + 17 \times 1 + 13 \times 2) / 4 = 16$

搜索空间访问的结点数为17



搜索空间



北京大学



# 影响算法效率的因素

最坏情况下的时间  $W(n) = (p(n)f(n))$

其中  $p(n)$  为每个结点时间， $f(n)$  为结点个数

影响回溯算法效率的因素

搜索树的结构

分支情况：分支均匀否

树的深度

对称程度：对称适合裁减

解的分布

在不同子树中分布多少是否均匀

分布深度

约束条件的判断：计算简单



北京大学



# 改进途径

根据树分支设计优先策略：

结点少的分支优先，解多的分支优先

利用搜索树的对称性剪裁子树

分解为子问题：

求解时间  $f(n)=c2^n$ ，组合时间  $T=O(f(n))$

如果分解为  $k$  个子问题，每个子问题大小为  $n/k$

求解时间为

$$kc2^{\frac{n}{k}} + T$$



北京大学



## 5.4 分支限界

组合优化问题的相关概念

目标函数（极大化或极小化）

约束条件

搜索空间中满足约束条件的解称为可行解

使得目标函数达到极大(或极小)的解称为最优解

### 5.4.1 背包问题

$$\max x_1 + 3x_2 + 5x_3 + 9x_4$$

$$2x_1 + 3x_2 + 4x_3 + 7x_4 \leq 10$$

$$x_i \in N, i = 1, 2, 3, 4$$



北京大学



# 分支限界技术(极大化)

## 设立代价函数

函数值以该结点为根的搜索树中的所有可行解的目标函数值的上界

父结点的代价不小于子结点的代价

## 设立界

代表当时已经得到的可行解的目标函数的最大值  
界的设定初值可以设为0

可行解的目标函数值大于当时的界，进行更新

## 搜索中停止分支的依据

不满足约束条件或者其代价函数小于当时的界



北京大学



# 实例：背包问题

背包问题的实例：

$$\begin{aligned}\max & x_1 + 3x_2 + 5x_3 + 9x_4 \\ & 2x_1 + 3x_2 + 4x_3 + 7x_4 \leq 10 \\ & x_i \in \mathbf{N}, i = 1, 2, 3, 4\end{aligned}$$

对变元重新排序使得

$$\frac{v_i}{w_i} \geq \frac{v_{i+1}}{w_{i+1}}$$

排序后实例

$$\begin{aligned}\max & 9x_1 + 5x_2 + 3x_3 + x_4 \\ & 7x_1 + 4x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_i \in \mathbf{N}, i = 1, 2, 3, 4\end{aligned}$$



北京大学





# 代价函数与分支策略确定

结点 $\langle x_1, x_2, \dots, x_k \rangle$  的代价函数

$$\sum_{i=1}^k v_i x_i + (b - \sum_{i=1}^k w_i x_i) \frac{v_{k+1}}{w_{k+1}}$$

若对某个  $j > k$  有  $b - \sum_{i=1}^k w_i x_i \geq w_j$

$$\sum_{i=1}^k v_i x_i$$

否则

分支策略----深度优先



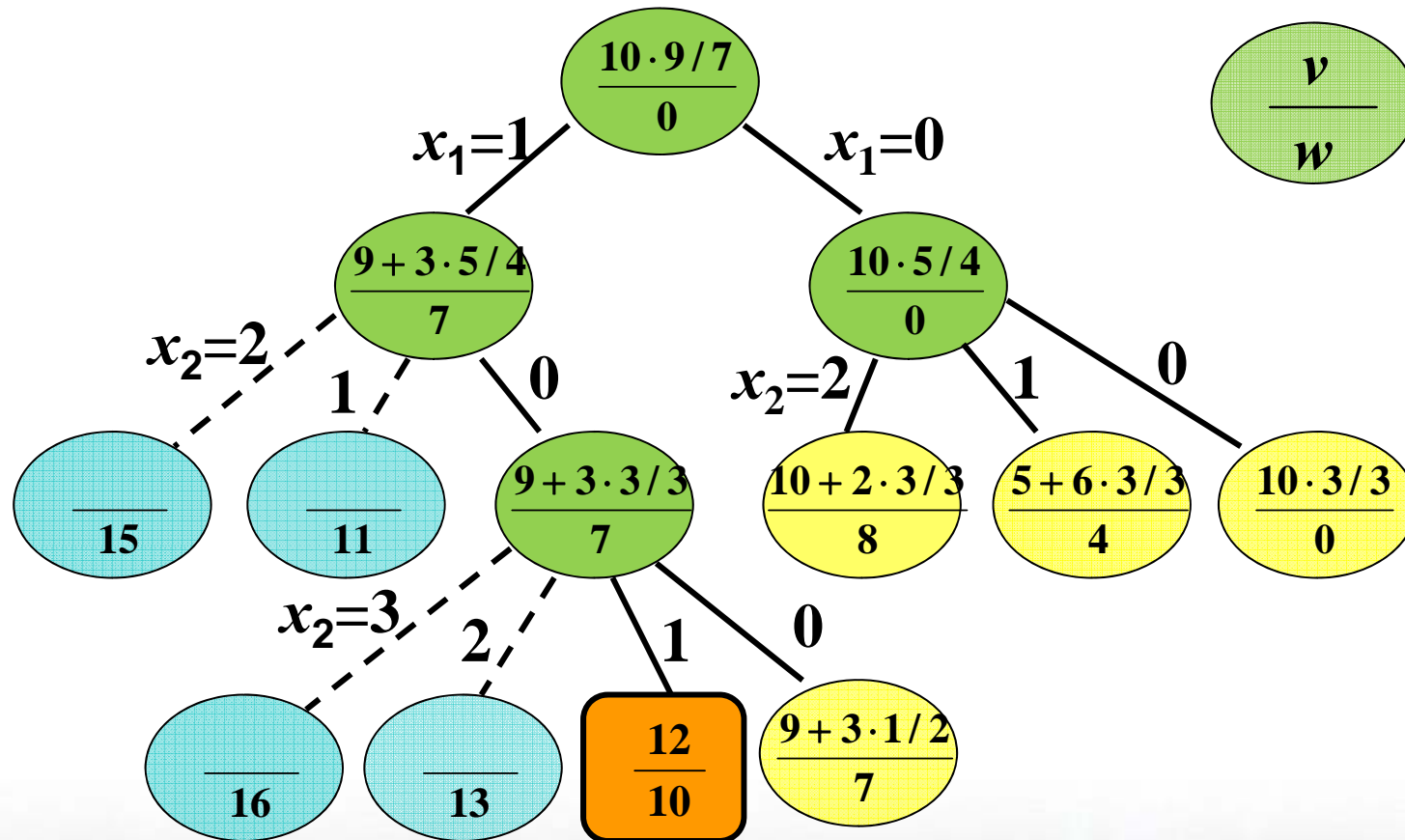
北京大学



$$\max 9x_1 + 5x_2 + 3x_3 + x_4$$

$$7x_1 + 4x_2 + 3x_3 + 2x_4 \leq 10, \quad x_i \in \mathbb{N}, i = 1, 2, 3, 4$$

实例



北京大学



## 5.4.2 最大团问题

问题：给定无向图  $G = \langle V, E \rangle$ , 求  $G$  中的最大团.

相关知识:

无向图  $G = \langle V, E \rangle$ ,

$G$  的子图:  $G' = \langle V', E' \rangle$ , 其中  $V' \subseteq V, E' \subseteq E$ ,

$G$  的补图:  $\check{G} = \langle V, E' \rangle$ ,  $E'$  是  $E$  关于完全图边集的补集

$G$  中的团:  $G$  的完全子图

$G$  的点独立集:  $G$  的顶点子集  $A$ , 且  $\forall u, v \in A, \{u, v\} \notin E$ .

最大团: 顶点数最多的团

最大点独立集: 顶点数最多的点独立集

命题:  $U$  是  $G$  的最大团当且仅当  $U$  是  $\check{G}$  的最大点独立集





# 算法设计

结点 $\langle x_1, x_2, \dots, x_k \rangle$ 的含义:

已检索  $k$  个顶点, 其中  $x_i=1$  对应的顶点在当前的团内  
搜索树为子集树

约束条件: 该顶点与当前团内每个顶点都有边相连

界: 当前图中已检索到的极大团的顶点数

代价函数: 目前的团扩张为极大团的顶点数上界

$$F = C_n + n - k$$

其中  $C_n$  为目前团的顶点数 (初始为0),

$k$  为结点层数

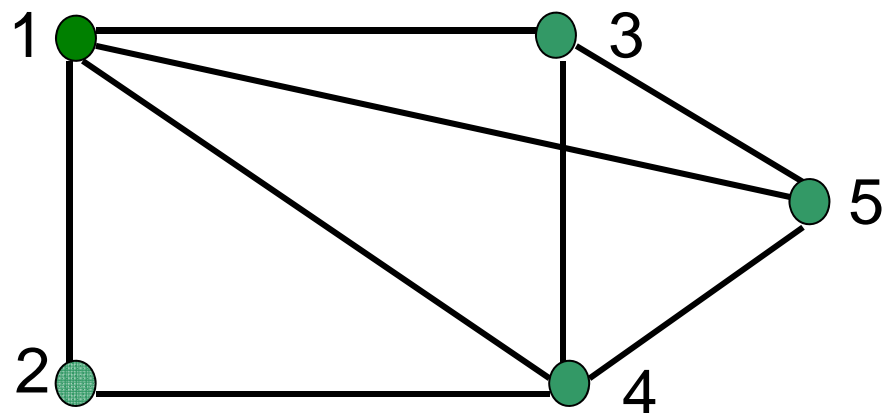
时间:  $O(n2^n)$



北京大學



# 最大团的实例



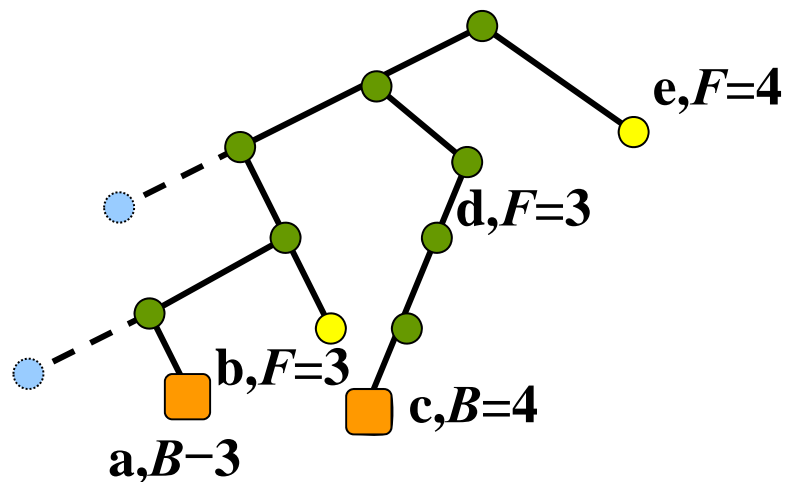
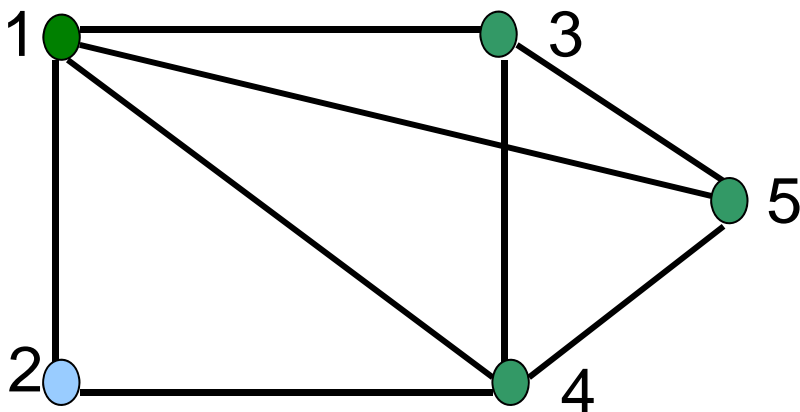
顶点编号顺序为 1, 2, 3, 4, 5,

对应  $x_1, x_2, x_3, x_4, x_5$ ,  $x_i=1$  当且仅当  $i$  在团内  
分支规定左子树为1, 右子树为0.

$B$  为界,  $F$  为代价函数值.



北京大学



- a: 得第一个极大团  $\{1, 2, 4\}$ , 顶点数为3, 界为3;
  - b: 代价函数值  $F = 3$ , 回溯;
  - c: 得第二个极大团  $\{1, 3, 4, 5\}$ , 顶点数为4, 修改界为4;
  - d: 不必搜索其它分支, 因为  $F = 4$ , 不超过界;
  - e:  $F = 4$ , 不必搜索.
- 最大团为  $\{1, 3, 4, 5\}$ , 顶点数为 4.





## 5.4.3 货郎问题

问题：给定 $n$ 个城市集合 $C=\{c_1, c_2, \dots, c_n\}$ , 从一个城市到另一个城市的距离 $d_{ij}$ 为正整数, 求一条最短且每个城市恰好经过一次的巡回路线.

货郎问题的类型：有向图、无向图.

设巡回路线从1开始,

解向量为 $\langle i_1, i_2, \dots, i_{n-1} \rangle$ ,

其中 $i_1, i_2, \dots, i_{n-1}$ 为 $\{2, 3, \dots, n\}$ 的排列.

搜索空间为排列树, 结点 $\langle i_1, i_2, \dots, i_k \rangle$ 表示得到 $k$ 步路线



北京大学



# 算法设计

**约束条件：** 令  $B = \{ i_1, i_2, \dots, i_k \}$ , 则

$$i_{k+1} \in \{ 2, \dots, n \} - B$$

**界：** 当前得到的最短巡回路线长度

**代价函数：** 设顶点  $c_i$  出发的最短边长度为  $l_i$ ,  $d_j$  为选定巡回路线中第  $j$  段的长度, 则

$$L = \sum_{j=1}^k d_j + \sum_{i_j \notin B} l_{i_j}$$

为部分巡回路线扩张成全程巡回路线的长度下界

时间  $O(n!)$ : 计算  $O((n-1)!)$  次, 代价函数计算  $O(n)$

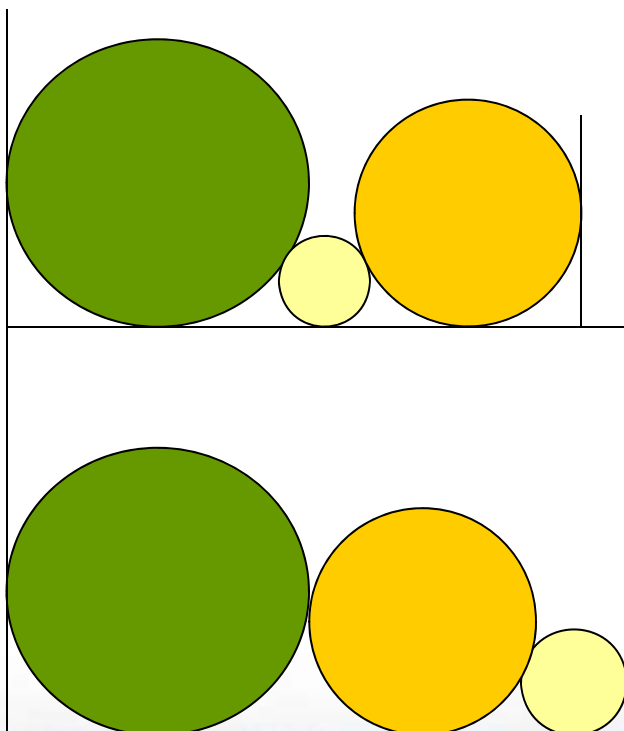


北京大学



## 5.4.4 圆排列问题

问题：给定 $n$ 个圆的半径序列，将各圆与矩形底边相切排列，求具有最小长度 $l_n$ 的圆的排列顺序。



解为 $\langle i_1, i_2, \dots, i_n \rangle$ 为 $1, 2, \dots, n$ 的排列，解空间为排列树。

部分解向量 $\langle i_1, i_2, \dots, i_k \rangle$ ：表示前 $k$ 个圆已排好。令 $B = \{i_1, i_2, \dots, i_k\}$ ，下一个圆选择 $i_{k+1}$ 。

约束条件： $i_{k+1} \in \{1, 2, \dots, n\} - B$

界：当前得到的最小圆排列长度



北京大学



# 代价函数符号说明

$k$ : 算法完成第  $k$  步, 已经选择了第1— $k$  个圆

$r_k$ : 第  $k$  个圆的半径

$d_k$ : 第  $k-1$  个圆到第  $k$  个圆的圆心水平距离,  $k>1$

$x_k$ : 第  $k$  个圆的圆心坐标, 规定  $x_1=0$ ,

$l_k$ : 第 1— $k$  个圆的排列长度

$L_k$ : 放好 1— $k$  个圆以后, 对应结点的代价函数值

$L_k \leq l_n$



北京大学



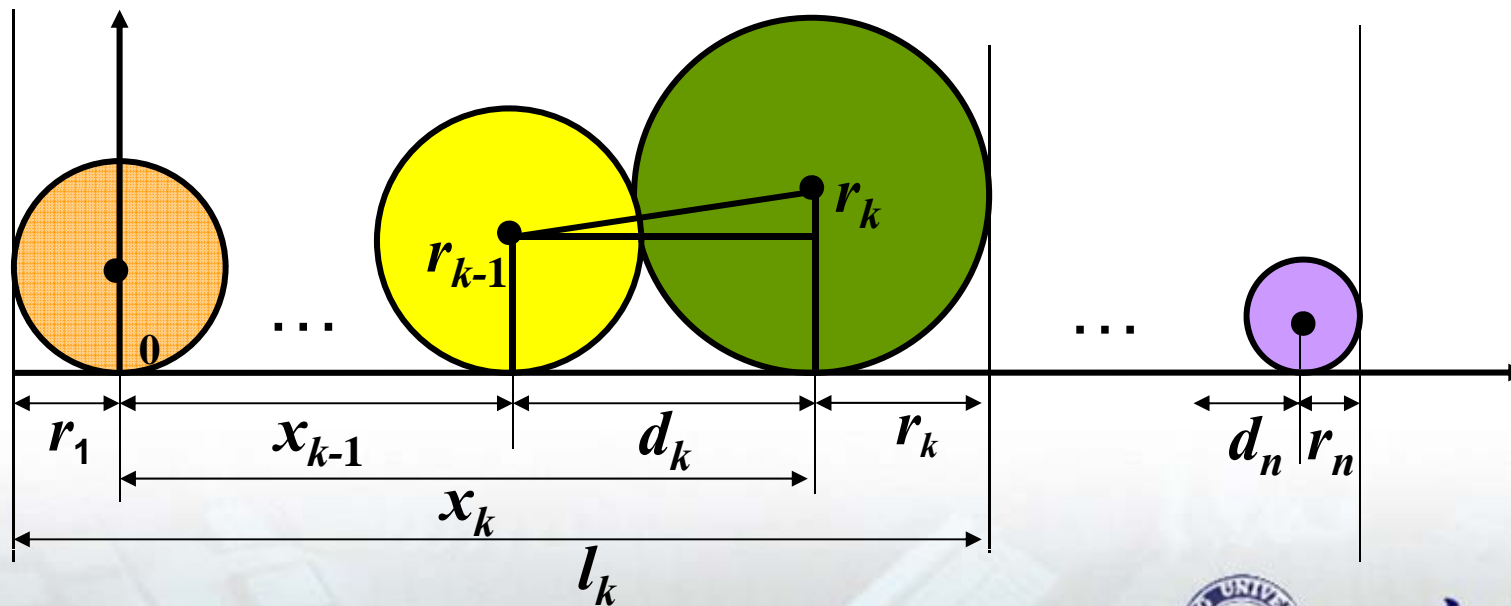
# 有关量的计算

$$d_k = \sqrt{(r_{k-1} + r_k)^2 - (r_{k-1} - r_k)^2} = 2\sqrt{r_{k-1}r_k}$$

$$x_k = x_{k-1} + d_k, \quad l_k = x_k + r_k + r_1$$

$$L_k = x_k + d_{k+1} + d_{k+2} + \dots + d_n + r_n + r_1$$

$$= x_k + 2\sqrt{r_k r_{k+1}} + 2\sqrt{r_{k+1} r_{k+2}} + \dots + 2\sqrt{r_{n-1} r_n} + r_n + r_1$$



北京大学



# 代价函数

排列长度是 $l_n$ ,  $L$ 是代价函数:

$$l_n = x_k + 2\sqrt{r_k r_{k+1}} + 2\sqrt{r_{k+1} r_{k+2}} + \dots + 2\sqrt{r_{n-1} r_n} + r_n + r_1$$

$$\geq x_k + 2(n-k)r + r + r_1$$

$$L = x_k + (2n - 2k + 1)r + r_1$$

$$r = \min(r_{i_j}, r_k) \quad i_j \in \{1, 2, \dots, n\} - B$$

$$B = \{i_1, i_2, \dots, i_k\},$$

时间:  $O(n n!)=O((n+1)!)$



北京大学





## 实例：计算过程

$$R = \{1, 1, 2, 2, 3, 5\}$$

取排列  $\langle 1, 2, 3, 4, 5, 6 \rangle$ ,

半径排列为：1, 1, 2, 2, 3, 5，结果见下表和下图

$k$	$r_k$	$d_k$	$x_k$	$l_k$	$L_k$
1	1	0	0	2	12
2	1	2	2	4	12
3	2	2.8	4.8	7.8	19.8
4	2	4	8.8	11.8	19.8
5	3	4.9	13.7	17.7	23.7
6	5	7.7	21.4	27.4	27.4



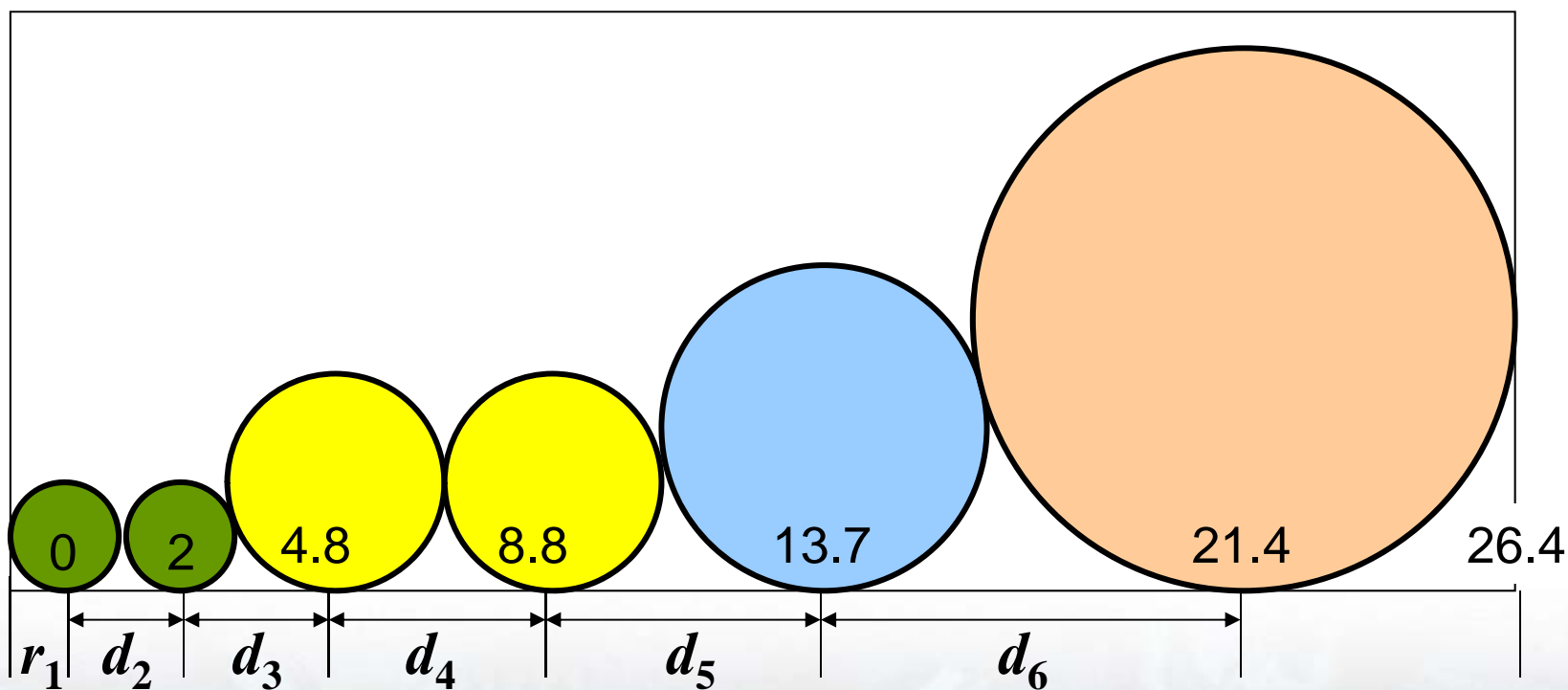


## 实例：图示

$$R = \{1, 1, 2, 2, 3, 5\}$$

取排列  $\langle 1, 2, 3, 4, 5, 6 \rangle$ , 半径排列为: 1, 1, 2, 2, 3, 5,

最短长度  $l_6 = 27.4$



北京大学



## 5.4.5 连续邮资问题

问题：给定 $n$ 种不同面值的邮票，每个信封至多 $m$ 张，试给出邮票的最佳设计，使得从1开始，增量为1的连续邮资区间达到最大？

实例： $n=5$ ， $m=4$ ，

面值  $X_1=\langle 1,3,11,15,32 \rangle$ ，邮资连续区间为 $\{1, 2, \dots, 70\}$

面值  $X_2=\langle 1,6,10,20,30 \rangle$ ，邮资连续区间为 $\{1, 2, 3, 4\}$

可行解： $\langle x_1, x_2, \dots, x_n \rangle$ ， $x_1=1$ ， $x_1 < x_2 < \dots < x_n$

约束条件：在结点 $\langle x_1, x_2, \dots, x_r \rangle$ 处，邮资最大连续区间为 $\{1, \dots, r_i\}$ ， $x_{i+1}$ 的取值范围是 $\{x_i+1, \dots, r_i+1\}$



北京大学



## $r_i$ 的计算

$y_i(j)$ : 用至多  $m$  张面值  $x_i$  的邮票加上  $x_1, x_2, \dots, x_{i-1}$  面值的邮票贴  $j$  邮资时的最少邮票数, 则

$$y_i(j) = \min_{1 \leq t \leq m} \{t + y_{i-1}(j - tx_i)\}$$

$$y_1(j) = j$$

$$r_i = \min\{j \mid y_i(j) \leq m, y_i(j+1) > m\}$$

搜索策略: 深度优先

界:  $max$ ,  $m$  张邮票可付的连续区间的最大邮资



北京大学





# 回溯算法小结

- (1) 适应于求解组合搜索问题（含组合优化问题）
- (2) 求解条件：满足多米诺性质
- (3) 解的表示：解向量，求解是不断扩充解向量的过程
- (4) 回溯条件：
  - 搜索问题-约束条件
  - 优化问题-约束条件+代价函数
- (5) 算法复杂性：最坏情况为指数，空间代价小
- (6) 降低时间复杂性的主要途径：
  - 利用对称性裁减子树
  - 划分成子问题
- (7) 分支策略（深度优先、宽度优先、宽深结合、优先函数）



北京大学