分支限界
及其应用

组合优化问题

- 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（解满足的条件）

  可行解: 搜索空间满足约束条件的解

  最优解: 使得目标函数达到极大 (或极小)的可行解

- 背包问题

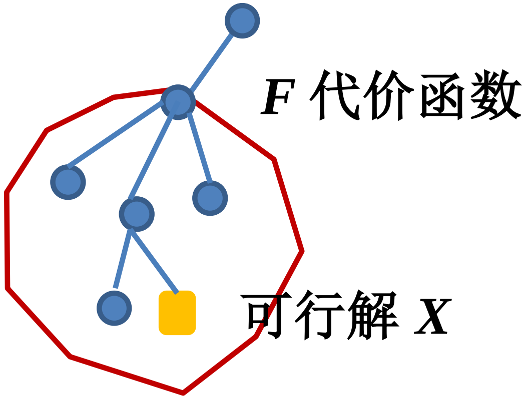
$$\max \quad x_1 + 3x_2 + 5x_3 + 9x_4$$

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

$$x_i \in \mathrm{N}, \quad i = 1,2,3,4$$

代价函数

- 计算位置：搜索树的结点
- 值：极大化问题是以该点为根的子树所有可行解的值的上界（极小化问题为下界）
- 性质：对极大化问题父结点代价不小于子结点的代价 (极小化问题相反)

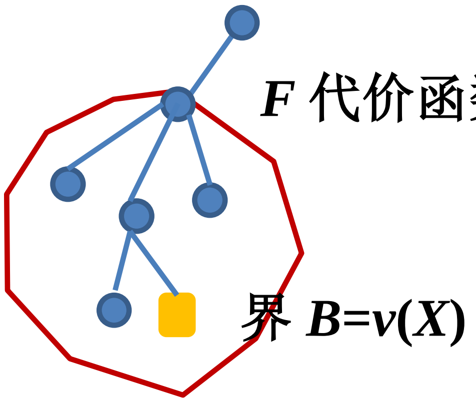


$F \geq v(X)$

极大化

界

- 含义：当前得到可行解的目标函数的最大值(极小化问题相反）
- 初值：极大化问题初值为 0 (极小化问题为最大值)
- 更新：得到更好的可行解时



分支限界

停止分支回溯父结点的依据：

1. 不满足约束条件

2. 对于极大化问题，代价函数值小于当前界（对于极小化问题是大于界）

界的更新

对极大化问题，如果一个新的可行解的优化函数值大于 (极小化问题为小于) 当前的界，则把界更新为该可行解的值

实例

背包问题：

4 种物品，重量 w_i 与价值 v_i 分别为

v_1=1, v_2=3, v_3=5, v_4=9

w_1=2, w_2=3, w_3=4, w_4=7

背包重量限制为10

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

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

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

代价函数的设定

- 对结点$< x_1, x_2, \ldots, x_k >$，估计以该结点为根的子树中可行解的上界.
- 按 v_i/w_i 从大到小排序，$i = 1,2, \ldots, n$
- 代价函数 = 已装入价值+ Δ

  Δ：还可继续装入最大价值的上界

  Δ = 背包剩余重量 × v_{k+1}/w_{k+1} (可装)

  Δ = 0 (不可装)

实例：背包问题

$$\max x_1 + 3x_2 + 5x_3 + 9x_4$$
$$2x_1 + 3x_2 + 4x_3 + 7x_4 \le 10$$
$$x_i \in \mathrm{N}, i = 1,2,3,4$$

对变元重新排序使得 $\dfrac{v_i}{w_i} \ge \dfrac{v_{i+1}}{w_{i+1}}$

排序后

$$\max 9x_1 + 5x_2 + 3x_3 + x_4$$
$$7x_1 + 4x_2 + 3x_3 + 2x_4 \le 10$$
$$x_i \in \mathrm{N}, i = 1,2,3,4$$

代价函数与分支策略

结点$<x_1, x_2, \ldots, x_k>$ 的代价函数

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

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

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

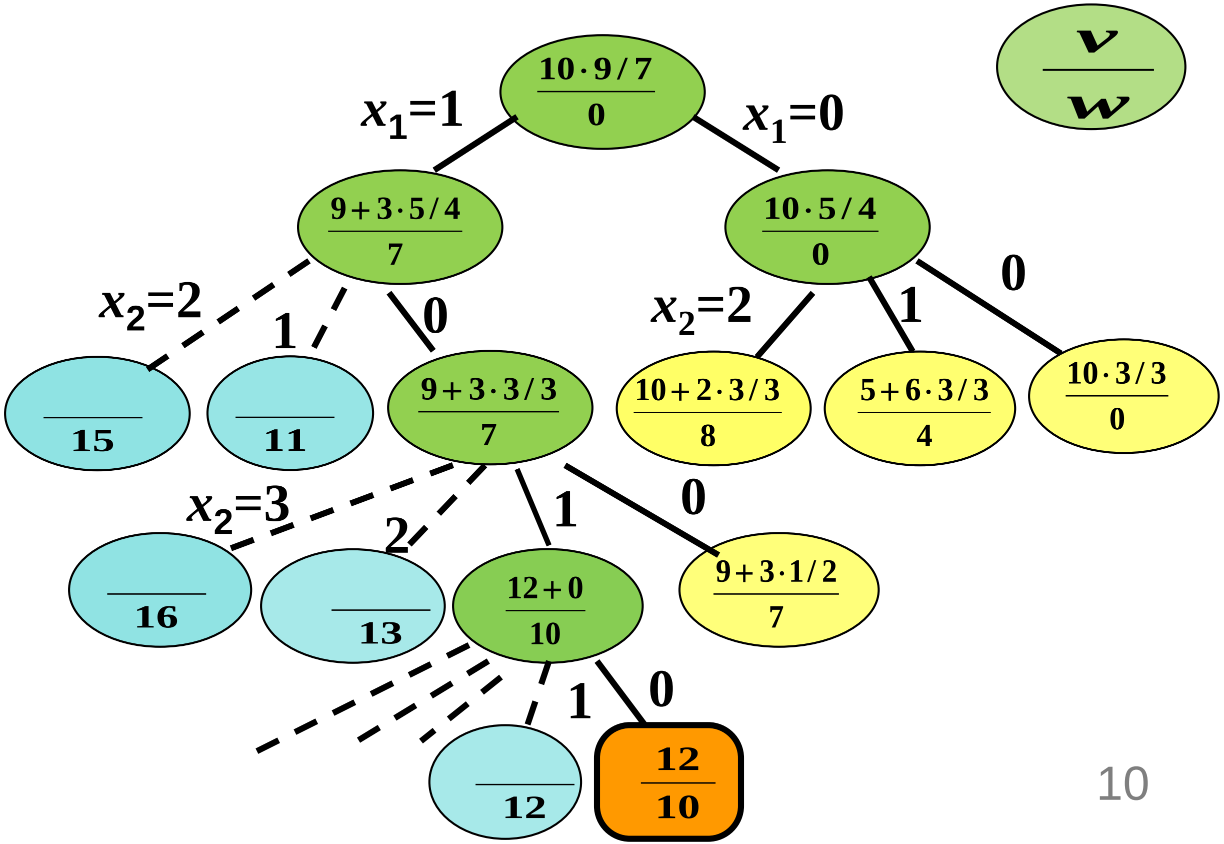
否则

分支策略----深度优先

实例

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

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



小结

- 分支限界适用于组合优化问题
- 对结点<x_1, ..., x_k>定义代价函数

  当前结点为根子树的可行解的上界或下界

  极大化问题与极小化问题的区别
- 定义界的初值

  得到新的更好的可行解就更新界

最大团问题

最大团问题

问题: 无向图G=<V,E>, 求G的最大团.

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

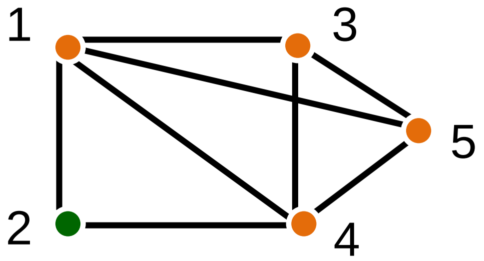
G的补图：$\overline{G} = \langle V, E' \rangle$, E'是E关于完全图边集的补集

G中的团：G 的完全子图

G的最大团：顶点数最多的团

实例

最大团：{ 1, 3, 4, 5 },



独立集与团

G 的点独立集：G 的顶点子集 A，且

$$\forall u, v \in A, \{u,v\} \notin E.$$

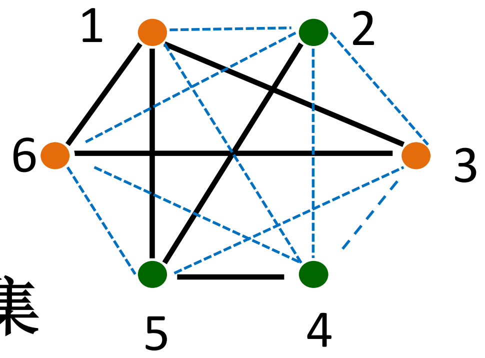
最大点独立集：顶点最多的点独立集

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

G 的最大团：

$U = \{ 1, 3, 6 \}$

补图 $\overline{G}$ 的最大点独立集



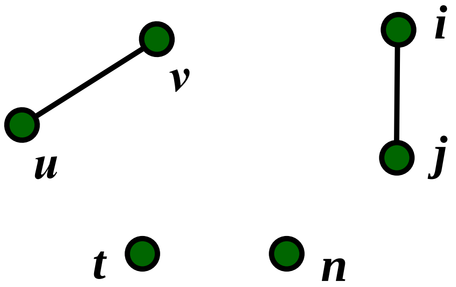
应用

编码, 故障诊断, 计算机视觉, 聚类分析, 经济学, 移动通信,VLSI电路设计,...

例子: 噪音使信道传输字符发生混淆

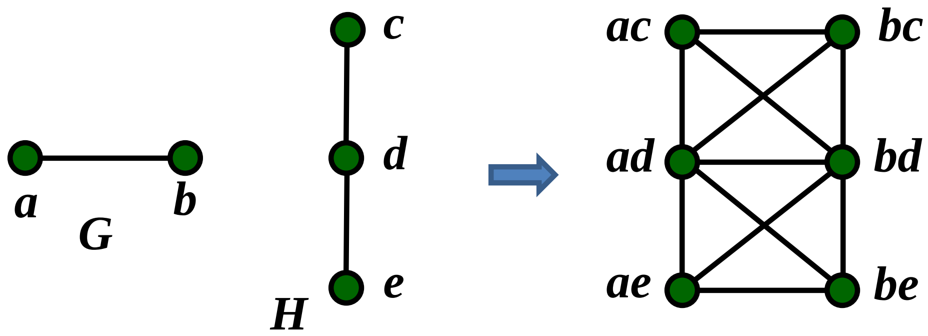
混淆图 $G=\langle V,E\rangle$， V为有穷字符集，

$\{u,v\}\in E \Leftrightarrow u$和$v$易混淆



编码设计

xy与uv混淆 $\Leftrightarrow$ x与u混淆且y与v混淆 $\vee$ $x=u$且y与v混淆$\vee$ x与u混淆且$y=v$



G与H 的正规积

为减少噪音干扰，设计代码应该找混淆图中的最大点独立集

最大团问题

问题：给定无向图 $G = < V, E >$, 其中顶点集 $V = \{ 1, 2, \ldots , n \}$，边集为 E .
求 G 中的最大团.

解： $< x_1, x_2, \ldots , x_n >$为 0–1向量，
x_k=1当且仅当顶点 k属于最大团.

蛮力算法：对每个顶点子集，检查是否构成团，即其中每对顶点之间是否都有边. 有 2^n 个子集，至少需要指数时间.

分支限界算法设计

搜索树为子集树.

结点 < $x_1, x_2, \ldots, x_k$ > 的含义：

已检索顶点 1, 2, … , k, 其中 x_i =1表示顶点 i 在当前的团内，i = 1, 2, … , k

约束条件：该顶点与当前团内每个顶点都有边相连

界：当前已检索到的极大团的顶点数

代价函数

代价函数：目前的团可能扩张为极大团的顶点数上界

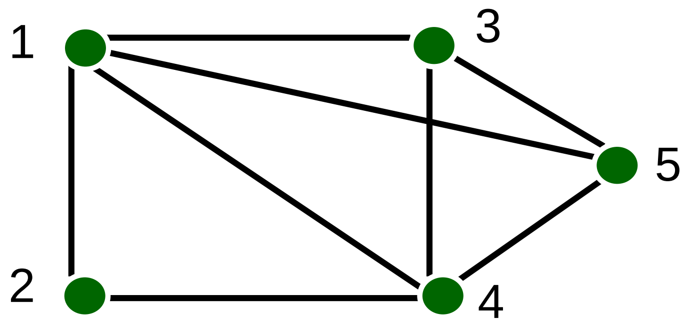
$$F = C_n + n - k$$

其中 C_n 为目前团的顶点数（初始为0）

k 为结点层数

最坏情况下时间：$O(n2^n)$

实例



顶点编号顺序为 1, 2, 3, 4, 5,

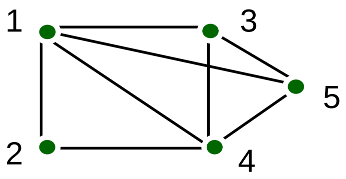
对应 x_1, x_2, x_3, x_4, x_5,

$x_i = 1$ 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.

搜索树



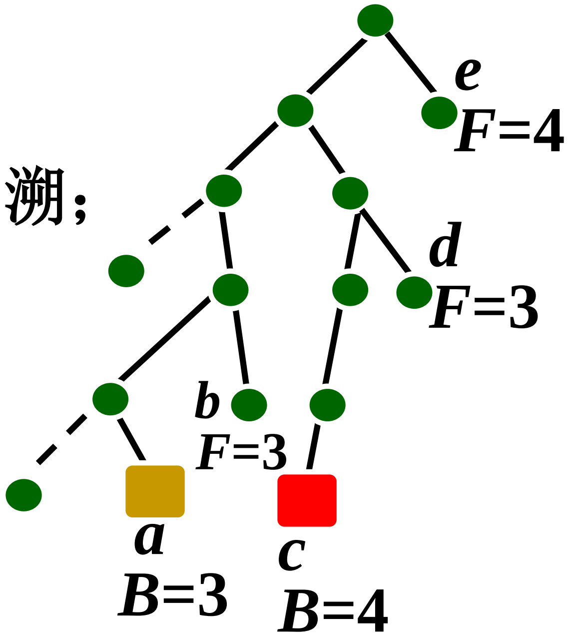
a: 极大团 {1,2,4},
顶点数为 3, 界 B=3;

b:代价函数值 F=3, 回溯;

c: 极大团 {1,3,4,5},
顶点数为 4,
修改界 B=4;

d: F=3,不必搜索;

e: F=4, 不必搜索.



输出最大团 {1,3,4,5}, 顶点数为 4.

小结

- 最大团问题的定义
  与点独立集的关系
- 分支限界算法的设计
  树的结构：子集树
  分支约束条件
  代价函数与界的设定
- 最坏情况的时间复杂度：$O(n2^n)$

货郎问题

货郎问题的定义

- 输入

  有穷个城市的集合 $C=\{c_1,c_2, \ldots, c_n\}$,

  距离 $d(c_i,c_j)=d(c_j,c_i)\in Z^+$, $1\le i<j\le n$

- 解：1,2,…,n 的排列 $k_1,k_2,\ldots,k_n$使得：

$$\min\{\sum_{i=1}^{n-1} d(c_{k_i},c_{k_{i+1}})+d(c_{k_n},c_{k_1})\}$$

算法设计

解向量为 $<1, i_1, i_2, \ldots, i_{n-1}>$, 其中 $i_1, i_2, \ldots, i_{n-1}$为 $\{2,3,\ldots,n\}$ 的排列.

搜索空间为排列树，结点$<i_1, i_2, \ldots, i_k>$表示得到 k 步路线 .

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

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

即每个结点只能访问一次 .

代价函数与界

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

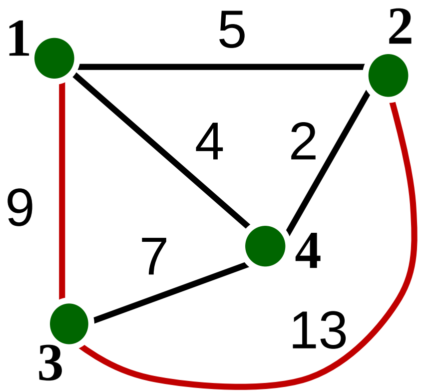
$$L = \sum_{j=1}^{k} d_j + l_{i_k} + \sum_{i_j \notin B} l_{i_j}$$

已走过路径长

剩余长度下界

代价函数

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



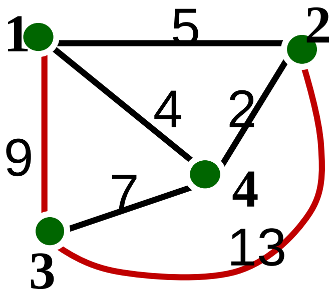
部分路线<1,3,2>

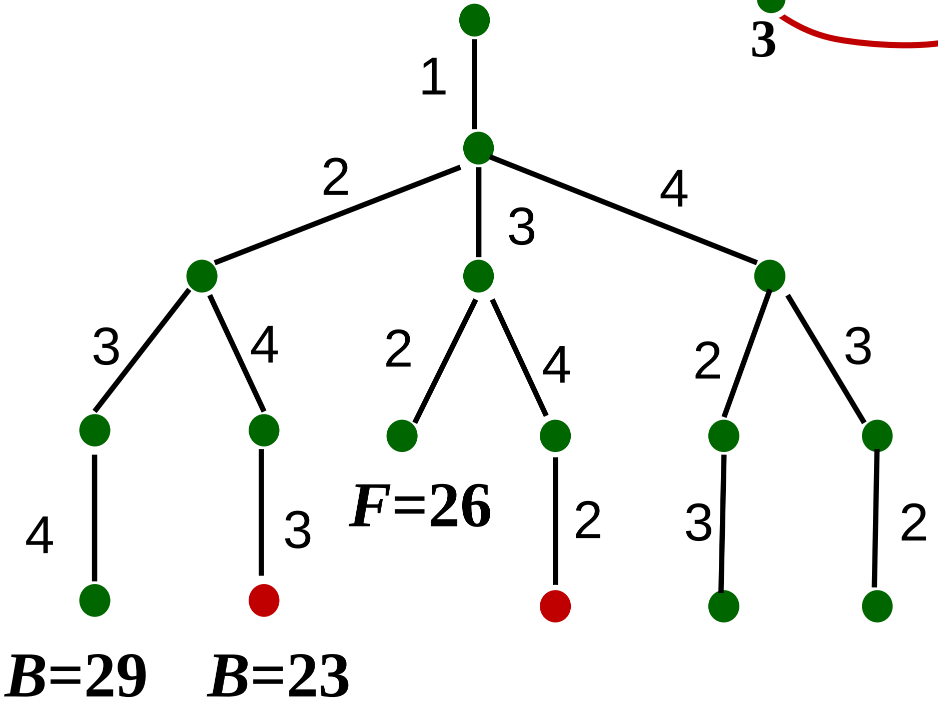
L=9+13+2+2=26

9+13为走过的路径长度

后两项分别为从结点2及4出发的最短边长

搜索树





实例运行

深度优先遍历搜索树

- 第一个界: <1,2,3,4>，长度为 29
- 第二个界: <1,2,4,3>，长度为 23
- 结点 <1,3,2>: 代价函数值 26>23，不再搜索, 返回<1,3>,右子树向下
- 结点<1,3,4>，代价函数值 9+7+2+2=20，继续，得到可行解 <1,3,4,2>，长度 23.
- 回溯到结点<1>，沿<1,4>向下...

最优解<1,2,4,3>或<1,3,4,2>，长度23

算法分析

- 搜索树的树叶个数：$O((n-1)!)$，每片树叶对应 1 条路径，每条路径有 n 个结点.
- 每个结点代价函数计算时间$O(1)$，每条路径的计算时间为 $O(n)$
- 最坏情况下算法的时间 $O(n!)$

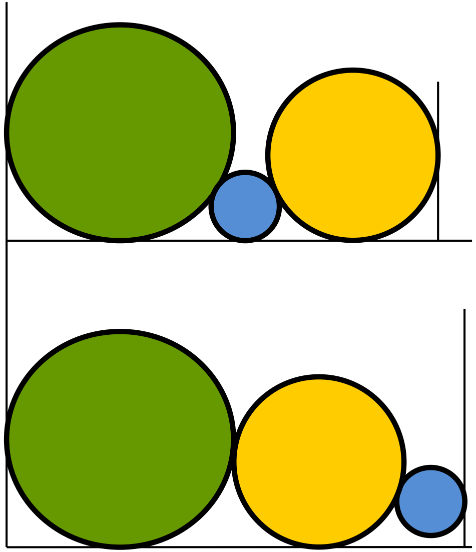
小结

- 货郎问题的分支限界算法：
  约束条件：只能选没有走过的结点
  代价函数：走过长度+后续长度的
  下界
- 时间复杂度：$O(n!)$

园排列问题

圆排列问题

给定n个圆的半径序列, 各圆与底线相切排列, 假定每个圆占大于1的长度, 求具有最小长度 l_n 的圆的排列顺序.



三个圆,半径给定

两种排列具有不同排列长度，第一种方式具有更小的排列长度

算法设计：分支限界

解：$<i_1,i_2,\ldots,i_n>$ 为 1,2,…,n 的排列

搜索空间为排列树

部分解向量 $<i_1,i_2,\ldots,i_k>$：表示前 k 个圆已排好. 令

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

下一个园选择 i_{k+1}

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

界：当前得到的最小园排列长度

符号说明

k： 算法已经选择了第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

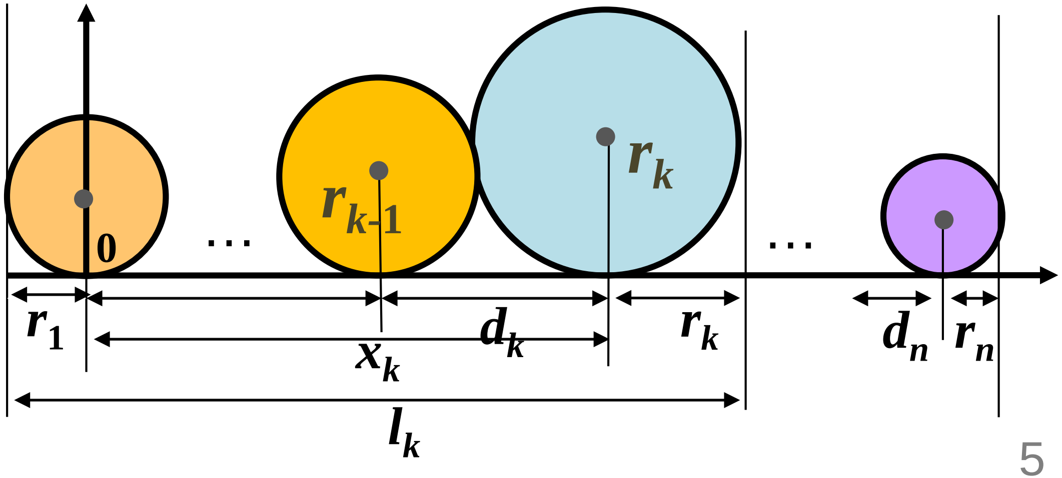
L_k： 放好 1—k 个圆以后，对应结点的代价函数值 $L_k \le l_n$

圆排列长度估计

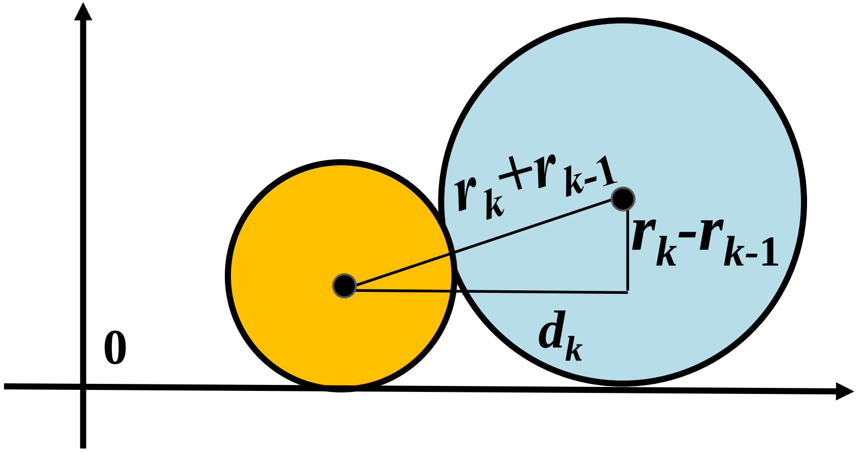
$x_k = x_{k-1} + d_k$

部分排列长度 $l_k = x_k + r_k + r_1$

排列长度 $l_n = x_k + d_{k+1} + d_{k+2} + \ldots + d_n + r_n + r_1$



有关量的计算



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

$$= 2\sqrt{r_{k-1} r_k}$$

代价函数

$$d_k = 2\sqrt{r_{k-1} r_k}$$

排列
长度

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

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

下界

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

$$r = \min(r_{i_j}, r_k)$$

$$i_j \in \{1, 2, \ldots, n\} - B$$

时间复杂度

- 搜索树树叶数为 $O(n!)$
- 每条路径代价函数的计算为 $O(n)$
- 最坏情况下算法时间复杂度为

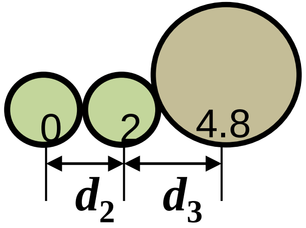
$$O(n\,n!) = O((n+1)!)$$

实例

输入： R={1,1,2,2,3,5}

排列　<1,2,3,4,5,6>

半径排列：1,1,2,2,3,5



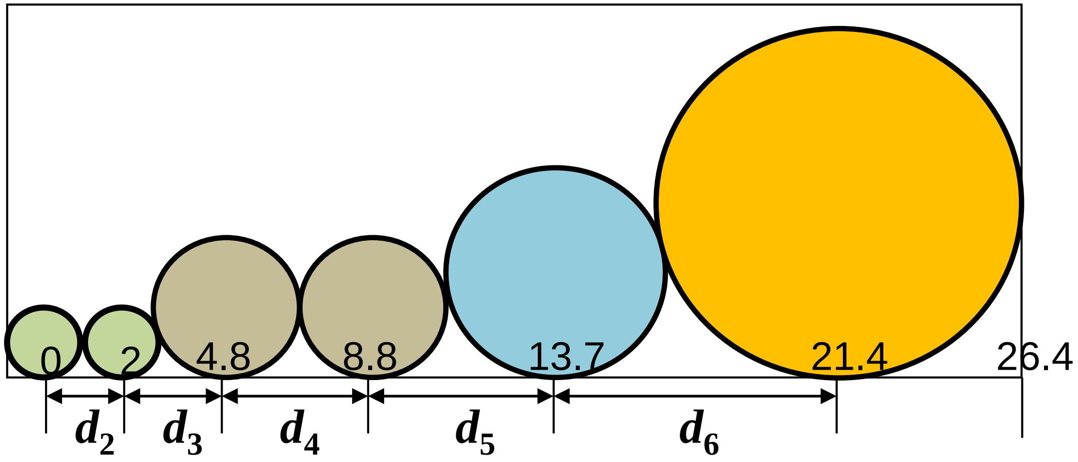
L_3=7.8+12=19.8

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

实例计算

半径排列：1,1,2,2,3,5，可行解l_6 =27.4

最优解：<1,3,5,6,4,2>, 最短长度26.5



小结

- 园排列问题的定义
- 如何设计代价函数？
  已有的排列长度+后续的圆排列
  长度的下界
- 时间复杂度估计：$O((n+1)!)$

连续邮资问题

连续邮资问题

问题：给定 n 种不同面值的邮票,每个信封至多贴 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：$n = 5$，$m = 4$.

设计1：面值 $X_1 = <1,3,11,15,32>$，
邮资连续区间 $\{1,2, \ldots ,70\}$

设计2：面值 $X_2 = <1,6,10,20,30>$，
邮资连续区间 $\{1,2,3,4\}$

算法设计

可行解： $< x_1,\ x_2, \ldots, x_n >$， $x_1 = 1$,

$x_1 < x_2 < \ldots < x_n$

搜索策略：深度优先

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

$$\{ x_i + 1, \ldots, r_i + 1 \}$$

若 $x_{i+1} > r_i + 1$， $r_i + 1$ 的邮资将没法支付.

r_i 的计算

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

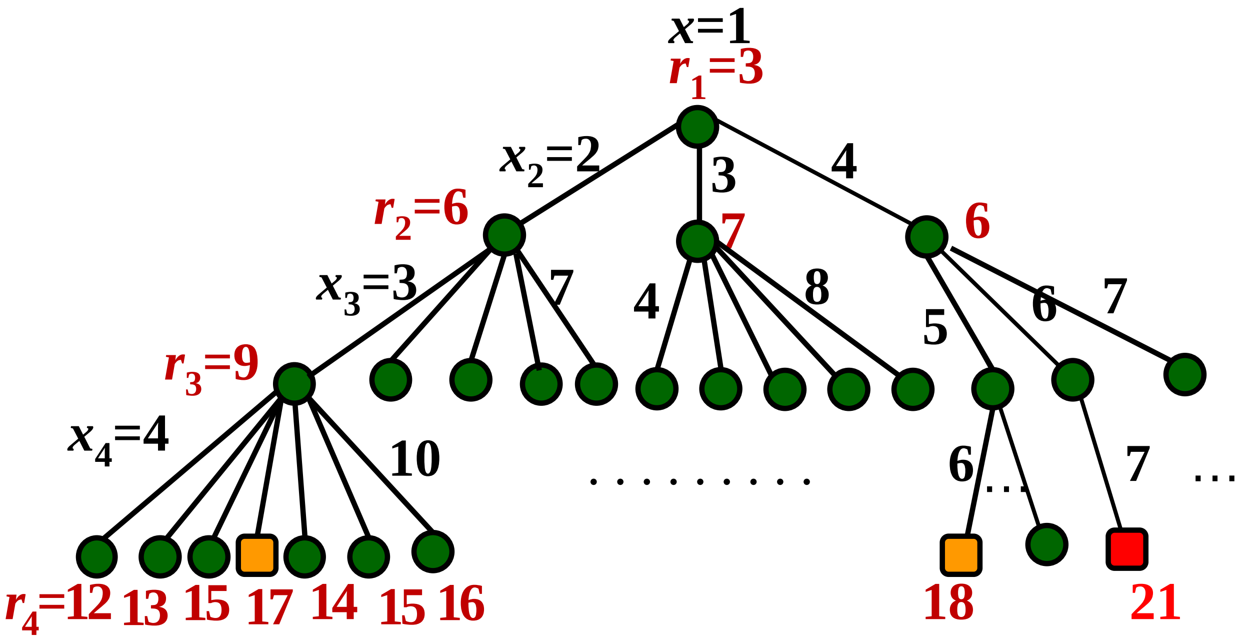
$$y_i(j) = \min_{1 \le t \le m}\{t + y_{i-1}(j - t\,x_i)\}$$

$$y_1(j) = j$$

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

界：max，m张邮票连续付的最大邮资

部分搜索树 n=4,m=3



解：X=<1,4,6,7>，最大连续区间 {1,…,21}

回溯算法小结

(1) 适于求解组合搜索问题及优化问题

(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) 分支策略：深度优先、宽度优先、宽深结合、函数优先

小结（续）

(6) 结点状态：白结点,黑结点,灰结点

(7) 算法时间复杂度：

$$W(n) = (\, p(n)\, f(n)\,)$$

其中 $p(n)$ 为每个结点的工作量

$f(n)$为结点个数

最坏情况下时间通常为指数级

平均情况下比蛮力算法好

空间代价小

小结（续）

(8) 降低时间复杂性的主要途径：

- 根据树的分支设计优先策略
  结点少分支优先，解多分支优先
- 利用搜索树的对称性裁减子树
- 分解为子问题，若求解时间$f(n)=c2^n$，组合时间$O(2^{n/k})$，分解为k个n/k规模子问题，则该算法时间为

$$T(n)=kc2^{n/k}+O(2^{n/k})=O(2^{n/k})=o(2^n)$$

算法设计与分析
课程总结

算法课程的知识框架

分治策略	动态规划	贪心法	回溯与分支限界	使用条件 主要步骤 分析方法 改进途径 典型例子

算法的数学基础：
序列求和、递推方程求解

算法的基本概念：
算法的伪码表示
算法的两种时间复杂度
复杂度函数阶的表示

函数的阶

- 阶的符号：$O, \Omega, \Theta, o, \omega$
- 阶的高低

  至少指数级：2^n, 3^n, $n!$, …

  多项式级：n, n^2, $n\log n$, $n^{1/2}$,…

  $\log n$的多项式级：$\log n$, $\log^2 n$,…

- 注意

  阶反映的是大的n（$n>n_0$）的情况
  可以忽略前面的有限项

序列求和

- 基本求和公式

  等比数列

  等差数列

  调和级数

- 估计和式的阶

  放大，然后估计上界

  用积分估计上下界

递推方程求解

- 主要的求解方法

  迭代 + 进行序列求和

  递归树 + 求和

  主定理：注意条件验证

- 一些常见的递推方程的解

$$f(n) = af(\frac{n}{b}) + d(n)$$

算法设计技术

- 设计技术
  分治策略
  动态规划
  贪心法
  回溯和分支限界
- 关注问题
  使用条件
  主要设计步骤
  时间复杂度分析方法
  改进途径
  典型例子

分治策略

- 适用条件：归约为独立求解子问题
- 设计步骤：归约方法, 初始子问题的计算, 子问题解的综合方法. 注意子问题划分均衡，类型相同
- 递归算法分析：求解递推方程
- 改进途径：减少子问题数, 预处理
- 典型问题：二分检索,归并排序,芯片测试,幂乘,矩阵乘法,最临近点对,多项式求值

动态规划

- 适用条件：优化问题, 多步判断求解, 满足优化原则,子问题重叠
- 设计步骤：确定子问题边界, 列关于目标函数的递推方程及初值；自底向上,备忘录存储；标记函数及解的追踪方法
- 复杂度分析：备忘录, 递推方程
- 典型问题:矩阵链相乘,投资,背包,最长公共子序列,图像压缩,最大子段和,最优二分检索树,生物信息学应用

贪心法

- 适用条件：组合优化问题, 多步判断求解, 有贪心选择性质
- 设计步骤：局部优化策略的确定及算法正确性证明 (直接证明, 数学归纳法,交换论证)
- 复杂度分析
- 典型问题：活动选择,装载问题, 最小延迟调度, 最优前缀码,最小生成树, 单源最短路

回溯和分支限界

- 适用条件：搜索或优化问题, 多步判断求解, 满足多米诺性质
- 设计步骤：确定解向量, 搜索树结构, 搜索顺序,结点分支搜索的约束条件与代价函数, 路径存储
- 搜索树结点数估计
- 复杂度分析
- 典型问题: *n*后问题, 背包问题,货郎问题,装载问题, 最大团问题, 圆排列问题, 连续邮资问题

算法设计

- 设计思想：尽量选复杂度低的算法
- 算法实现依赖于数据结构，选择合适的数据结构
- 实际问题中的综合考虑：时空权衡，实现成本的权衡，...