

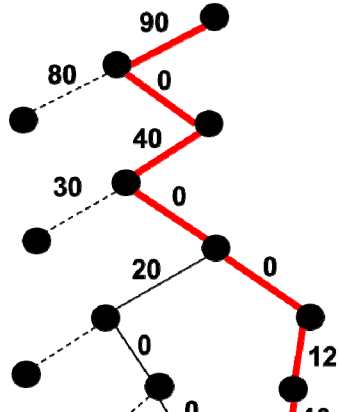


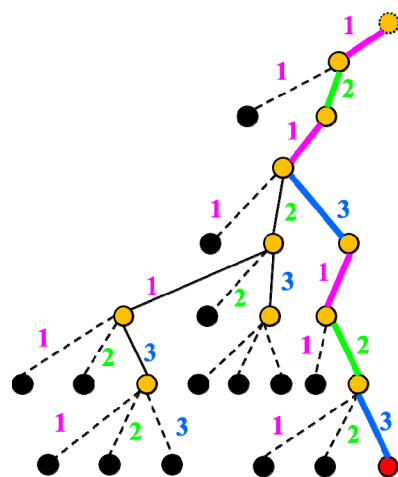
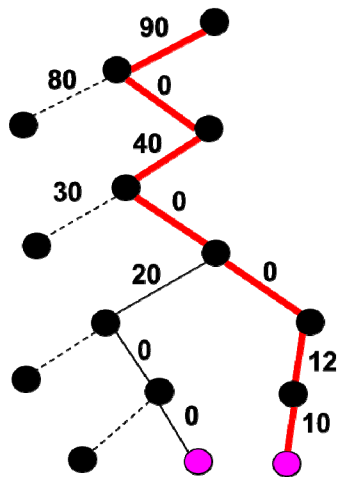
算法分析与设计

Analysis and Design of Algorithm

Lesson 14

要点回顾

- 回溯法适用条件
 - 多米诺性质及其作用
 - 回溯法两种实现
 - 递归实现（回溯最优子结构性性质）
 - 迭代实现（回溯贪心选择性性质）
 - 回溯法几个实例
 - 装载问题
 - 图着色问题
 - 分支限界法（开端）
- 



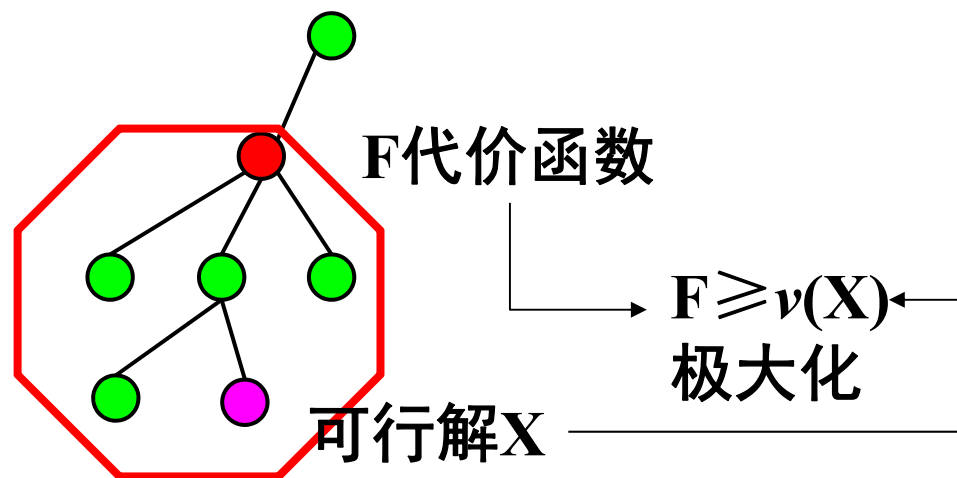


分支限界法

- 分支限界：一种与回溯法类似的算法
 - 将问题建模为搜索解空间树
 - 通常用代价函数估算每个分支的最优值
 - 优先选择当前看来最好的分支
 - 搜索策略一般采用宽度优先搜索
 - 搜索过程中剪枝
- 分支限界的剪枝函数
 - 不满足约束条件
 - 代价函数值不优于当前的界

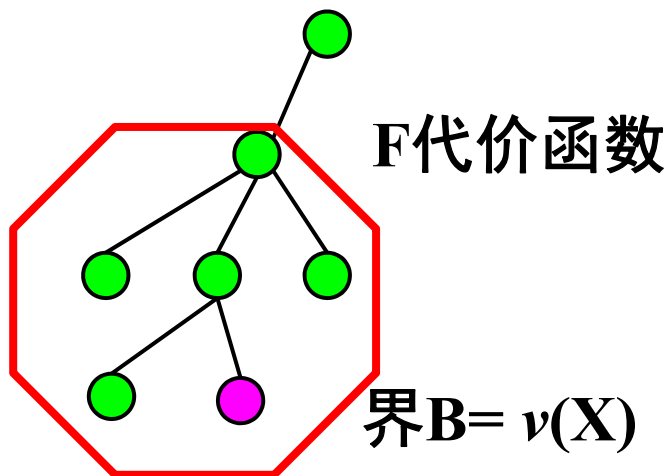
代价函数

- **计算位置：** 搜索树的结点
- **估值：** 极大化问题是以该点为根的子树所有可行解的值的**上界**（极小化问题则为下界）
- **性质：** 对极大化问题父节点代价**不小于**子结点的代价（极小化问题则相反）



界

- **含义：**当前得到可行解的**目标函数**最大值（极小化问题则相反）
- **初值：**极大化问题初值为0（极小化问题则为最大值）
- **更新：**得到更好的可行解时



界 \subseteq 代价函数，即： $F \geq B$



分支限界

- 停止分支回溯父节点的依据

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

分支限界解组合优化问题

■ 背包问题

背包限重为10

| 物品 <i>i</i> 属性 | 1 | 2 | 3 | 4 |
|-------------------|---|---|---|---|
| 价值 v_i | 1 | 3 | 5 | 9 |
| 重量 w_i | 2 | 3 | 4 | 7 |

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

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



代价函数的设定

- 对结点 $\langle x_1, x_2, \dots, x_k \rangle$, 估计以该结点为根的子树中可行解的上界
- 按单位重量的价值 v_i / w_i 从大到小排序
- 代价函数=已装入价值+ Δ
 - Δ : 还可以继续装入最大价值的上界
 - $\Delta = \text{背包剩余重量} \times v_{k+1} / w_{k+1}$ (可装)
 - $\Delta = 0$ (不可装)



实例：背包问题

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

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

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

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

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



代价函数与分支策略

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

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

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

否则 $F = \sum_{i=1}^k v_i x_i$

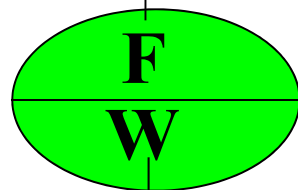
分支策略——深度优先+代价函数优先

实例

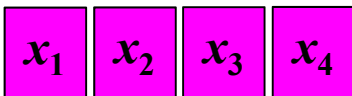
$$\max 9x_1 + 5x_2 + 3x_3 + x_4$$

$$s.t. \begin{cases} 7x_1 + 4x_2 + 3x_3 + 2x_4 \leq 10 \\ x_i \in \mathbb{N}, i = 1, 2, 3, 4 \end{cases}$$

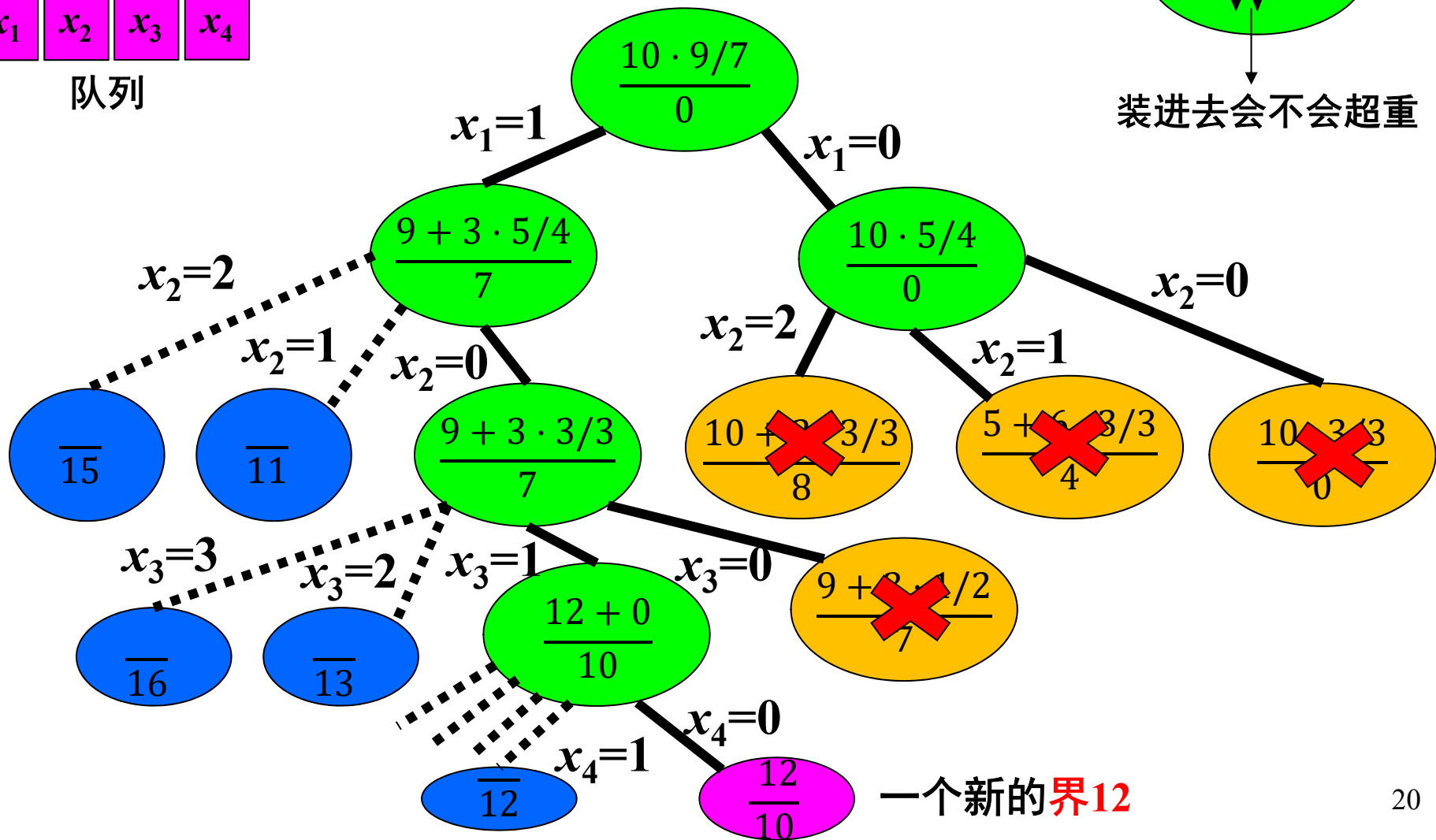
代价函数计算的值



装进去会不会超重



队列



一个新的界12

0-1背包问题



■ 实例

- 4种物品，重量 w_i 和价值 v_i 分别为
- $v_1 = 1, v_2 = 3, v_3 = 5, v_4 = 10$
- $w_1 = 2, w_2 = 3, w_3 = 6, w_4 = 7$
- 背包重量限制为10

■ 建模：

最大化 $x_1 + 3x_2 + 5x_3 + 10x_4$

满足约束条件
$$\begin{cases} 2x_1 + 3x_2 + 6x_3 + 7x_4 \leq 10 \\ x_i \in \{0,1\}, \quad i = 1, 2, 3, 4 \end{cases}$$



0-1背包问题—代价函数

- 按 v_i/w_i 从大到小排序, $i = 1, 2, \dots, n$
- 假设位于结点 $\langle x_1, x_2, \dots, x_k \rangle$
- 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 $\times v_{k+1}/w_{k+1}$ (可装)
 - $\Delta=0$ (不可装)

0-1背包问题—分支限界法

■ 基本思想

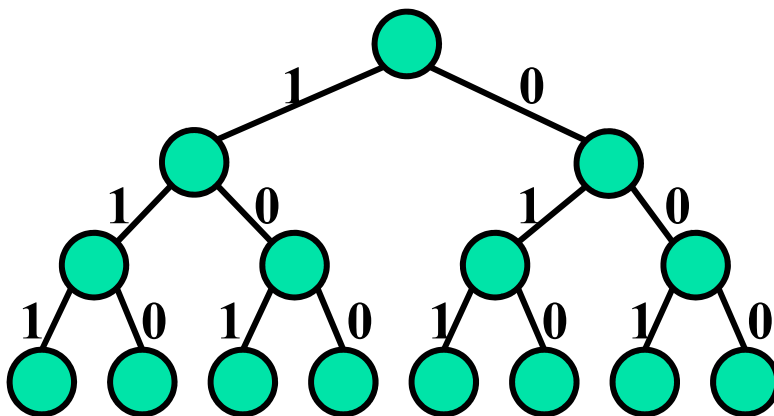
1. 将物品按 v_i/w_i 从大到小排序，确定解空间树
2. 从空集 \emptyset 和仅含空集 \emptyset 的优先队列开始
3. 选择计算节点队列中代价值最高的节点并扩展
4. 若扩展出节点不被剪枝，将节点插入节点队列
5. 反复2~3步，直到优先队列为空时为止

■ 代价函数

- 已装入价值 $+\Delta$

■ 剪枝函数

- 与回溯法相同



0-1背包问题—分支限界法

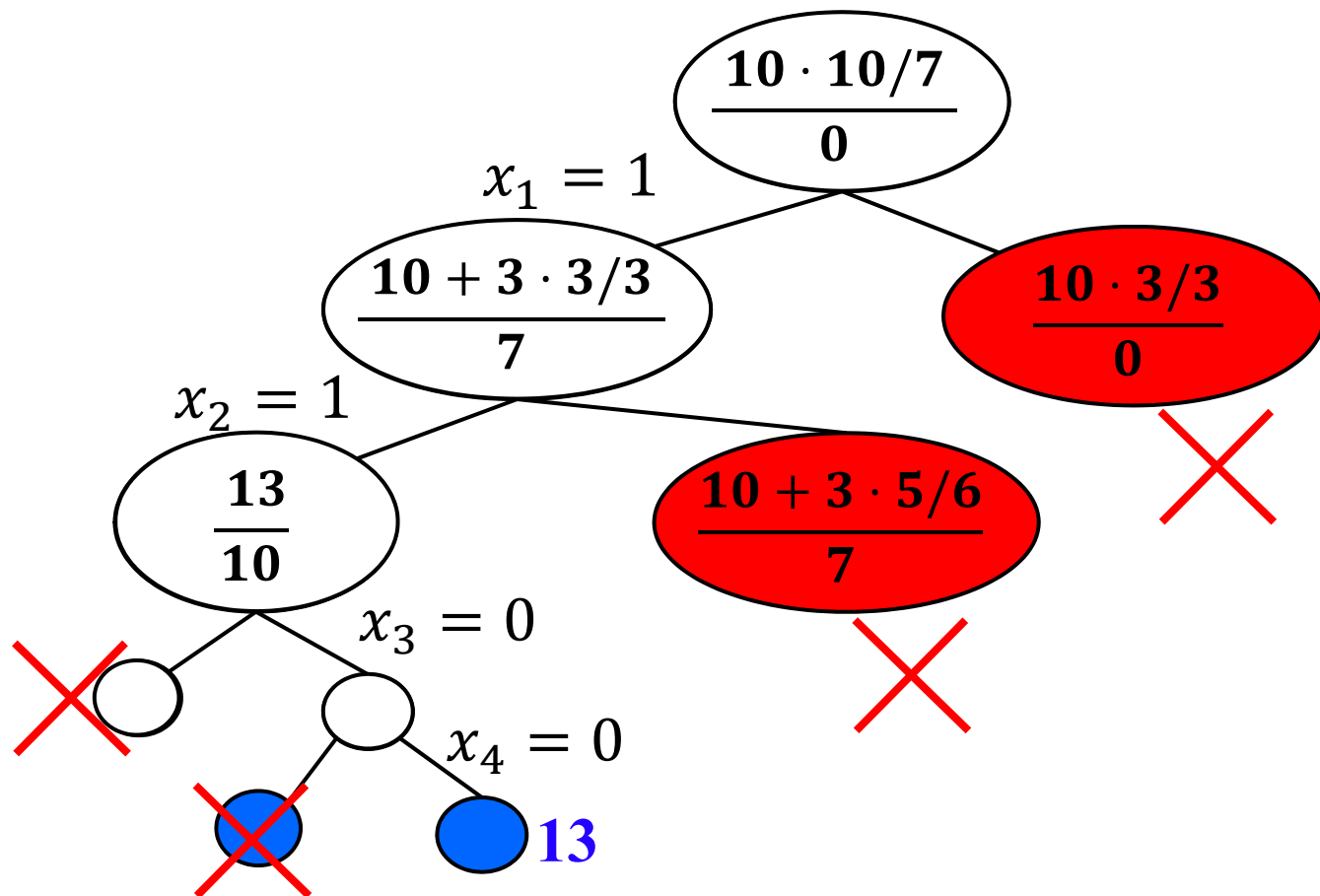
最大化 $10x_1 + 3x_2 + 5x_3 + x_4$

满足 $7x_1 + 3x_2 + 6x_3 + 2x_4 \leq 10; x_i \in \{0,1\}, i = 1, 2, 3, 4$

代价函数计算的值

$$\frac{F}{W}$$

装进去会不会超重





TSP问题

输入： 城市集 $C=\{c_1, c_2, \dots, c_n\}$, 距离
 $d(c_i, c_j)=d(c_j, c_i)$

解： $1, 2, \dots, n$ 的排列 k_1, k_2, \dots, k_n 使得

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



算法设计

- **解向量**为: $\langle 1, i_1, i_2, \dots, i_{n-1} \rangle$, 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
- **约束条件**: 令 $O = \{i_1, i_2, \dots, i_k\}$ 则 $i_{k+1} \in \{2, 3, \dots, n\} - O$, 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

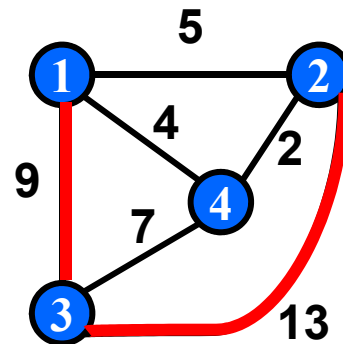
代价函数

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

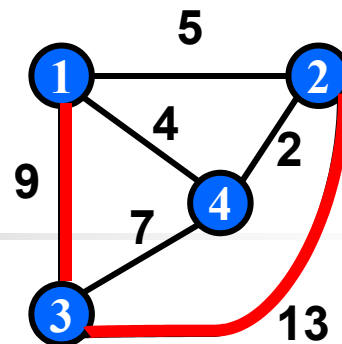
部分路线<1, 3, 2>

$$L=9+13+2+2=26$$

- 9+13为走过的路径长度
- 后两项分别为从结点2及结点4出发的最短边长

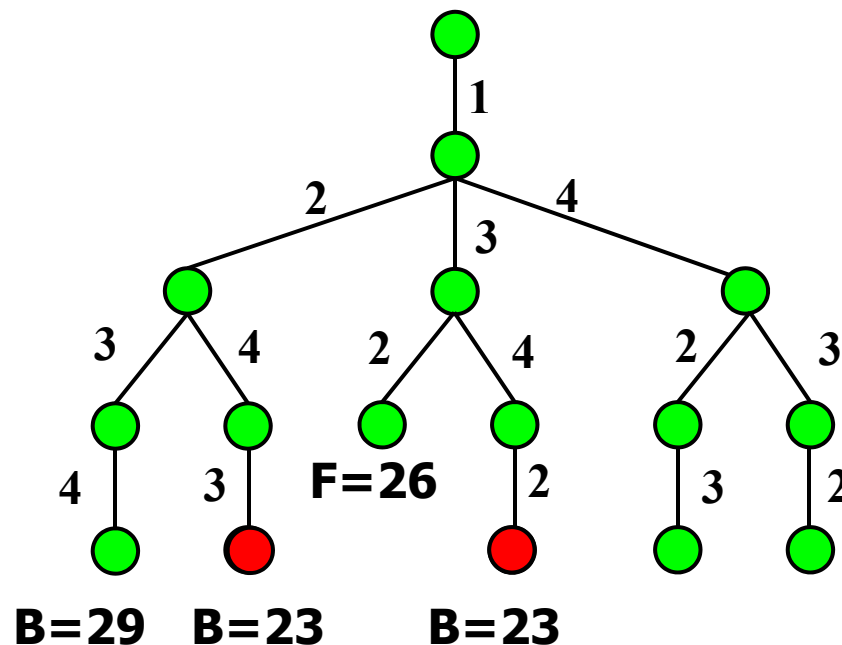


搜索树



深度优先遍历搜索树

- 第1个界: $\langle 1, 2, 3, 4 \rangle$, $B=29$
- 第2个界: $\langle 1, 2, 4, 3 \rangle$, $B=23$
- 结点 $\langle 1, 3, 2 \rangle$: 代价函数值 $26 > 23$, 不再搜索, 返回 $\langle 1, 3 \rangle$, 右子树向下
- 结点 $\langle 1, 3, 4 \rangle$: 代价函数值 $9 + 7 + 2 + 2 = 20 < 23$, 继续, 得到可行解 $\langle 1, 3, 4, 2 \rangle$, 长度23
- 回溯到结点 $\langle 1 \rangle$, 沿 $\langle 1, 4 \rangle$ 向下
- ... **→ 最优解:** $\langle 1, 2, 4, 3 \rangle$ 或 $\langle 1, 3, 4, 2 \rangle$, 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，每条路径有个 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 $O(n)$
- 最坏情况下算法的时间 $O(n!)$

一个关于巡回演唱会的例子

■ 薛之谦2019演唱会

- **地点：**北京、上海、广州、深圳、南京、杭州、武汉、成都、重庆、雄安
- **线路：**从北京出发，跑遍各大城市，回到北京



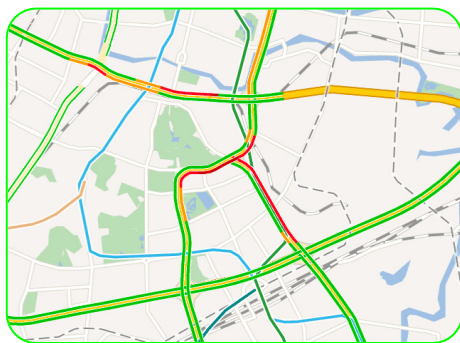
- **目标：**考虑机票价格，确定票价最少的线路

| 票价 | 北京 | 上海 | 广州 | |
|-------|-------|-------|-------|-------|
| 北京 | 0 | 500 | 600 | |
| 上海 | 100 | 0 | 800 | |
| 广州 | 1000 | 200 | 0 | |
| | | | | |

非对称旅行商问题

■ 问题定义

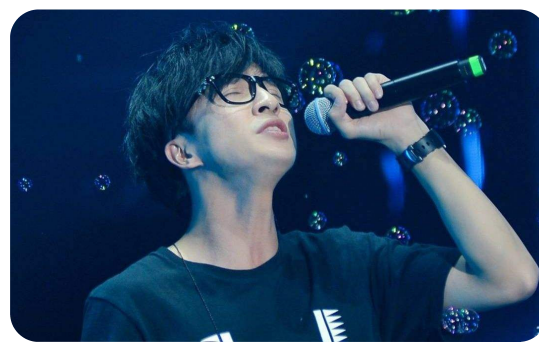
- 城市集合: $C = \{c_1, c_2, \dots, c_n\}$
- 城市距离: $d(c_i, c_j)$
- 距离不对称: $d(c_i, c_j) \neq d(c_j, c_i)$
- **目标:** 求遍历所有城市（不重复）的最短路径



道路拥堵情况下的
送快递问题



考虑城市单行线
的送快递问题

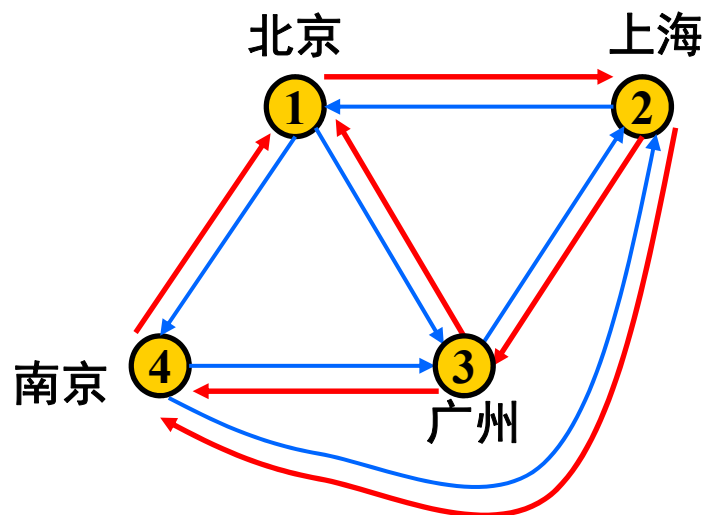


全国巡回演唱会的
路线安排问题

非对称旅行商问题的解空间

■ 实例

| 票价 | 北京 | 上海 | 广州 | 南京 |
|----|------|-----|-----|------|
| 北京 | 0 | 500 | 600 | 100 |
| 上海 | 100 | 0 | 800 | 500 |
| 广州 | 1000 | 200 | 0 | 2000 |
| 南京 | 400 | 400 | 100 | 0 |



■ 最优解

- 解的表示: $\langle 1, 4, 3, 2 \rangle$
- 路线: 北京 \rightarrow 南京 \rightarrow 广州 \rightarrow 上海 \rightarrow 北京
- 总票价: $100 + 100 + 200 + 100 = 500$

非对称旅行商问题的解空间树

| | 1 | 2 | 3 | 4 |
|---|------|-----|-----|------|
| 1 | 0 | 500 | 600 | 100 |
| 2 | 100 | 0 | 800 | 500 |
| 3 | 1000 | 200 | 0 | 2000 |
| 4 | 400 | 400 | 100 | 0 |

回溯法

- 深度优先遍历解空间树
- 剪枝函数：比较当前解与当前最优解

