分支限界算法

陈长建

计算机科学系

第四次上机实验

- **院楼103，12月21日**上午8:30-12:00（现场验收）
- 实验离线题（离线准备）
  - 回溯算法实现题5-4运动员最佳配对问题
  - 分支限界法求解实现题6-3无向图的最大割问题
- 在线题
  - acm.hnu.edu.cn

最大/小堆（heap）

- 堆的复杂度
  - 构建: ?
  - 插入: ?

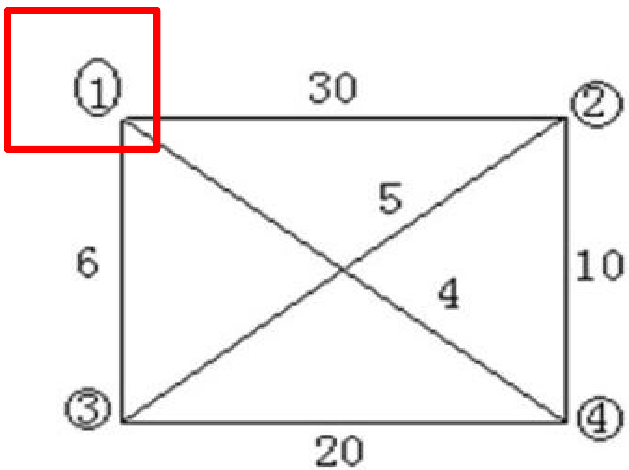
回顾

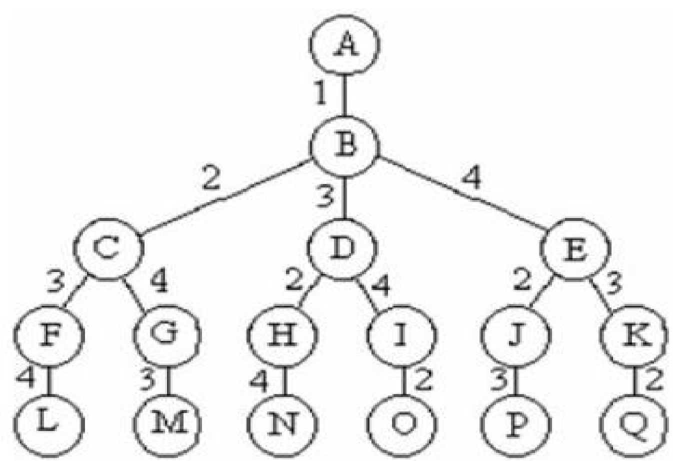
- 一旦有一个叶结点成为当前扩展结点，则可以断言该叶结点所相应的解即为最优解。此时可终止算法。

- 3个集装箱， 重量分别为 5, 20, 10，两艘船载重量分别为C1=30，C2=45。请分别用队列式分支限界法和优先队列式分支限界法求解该问题，并写出相应的节点序列。并对比其异同。

分支限界法案例3- 旅行商（TSP）问题

- 某售货员要到若干城市去推销商品，已知各城市之间的路程（旅费），他要选定一条从驻地出发，经过每个城市一遍，最后回到驻地的路线，使总的路程（总旅费）最小。





算法描述（1/4）-初始化

- 创建一个最小堆表示活结点优先队列。堆中每个结点的子树费用的下界 lcost 值是优先队列的优先级。
- 计算出图中每个顶点的最小费用出边并用 minout 记录。如果所给的有向图中某个顶点没有出边，则该图不可能有回路，算法即告结束。
- 如果每个顶点都有出边，则根据计算出的 minout 作算法初始化。

算法描述（2/4）-内部节点扩展

- 叶节点的父节点： s=n-2。如果该叶结点相应一条可行回路且费用小于当前最小费用，则将该叶结点插入到优先队列中，否则舍去该叶结点。

算法描述（3/4）-内部节点扩展

- 更上层节点：s<n-2，依次产生当前扩展结点的所有子结点。
  - 由于当前扩展结点所相应的路径是x[0:s]，其可行子结点是从剩余顶点x[s+1:n-1]中选取的顶点x[i]，且(x[s]，x[i])是所给有向图G中的一条边。对于当前扩展结点的每一个可行子结点，计算出其前缀(x[0:s], x[i])的费用cc和相应的下界lcost。
  - 当lcost<bestc时，将这个可行儿子结点插入到活结点优先队列中 下界/优先级： lcost=cc+sum(minout)

算法描述（4/4）- 循环终止条件

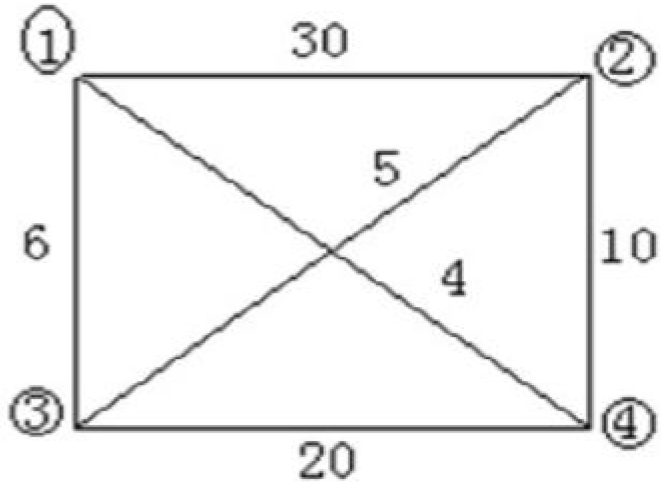
- 排列树的一个叶结点成为当前扩展结点。当s=n-1时，已找到的回路前缀是x[0:n-1]，它已包含图G的所有n个顶点。因此，当s=n-1时，相应的扩展结点表示一个叶结点。

算法结束时的效果

- 第一条到达叶节点的即最优解。
- 此时该叶结点所相应的回路的费用等于cc和lcost的值。
- 剩余的活结点的lcost值不小于已找到的回路的费用。它们都不可能导致费用更小的回路。因此已找到的叶结点所相应 的回路是一个最小费用旅行售货员回路，算法可以结束。
- 算法结束时返回找到的最小费用，相应的最优解由数组v 给出。

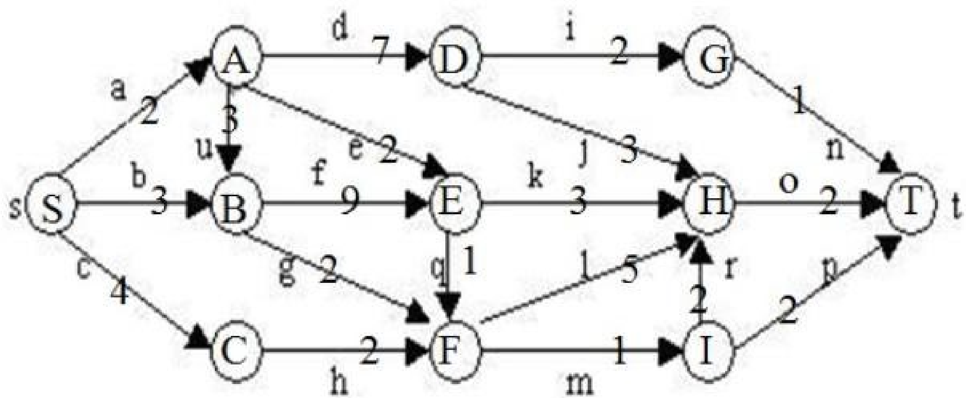
练习

- 优先级队列式分支限界法 求解如下TSP问题



分支限界法案例4- 单源最短路径问题

- 在下图所给的有向图G中，每一边都有一个非负边权。要求图G的从源顶点s到目标顶点t之间的最短路径。
- 输入：一个有向加权连通图G = <V,E,w>
- 输出：从源点s到终点t的最短路径
- 约束条件：路径最短

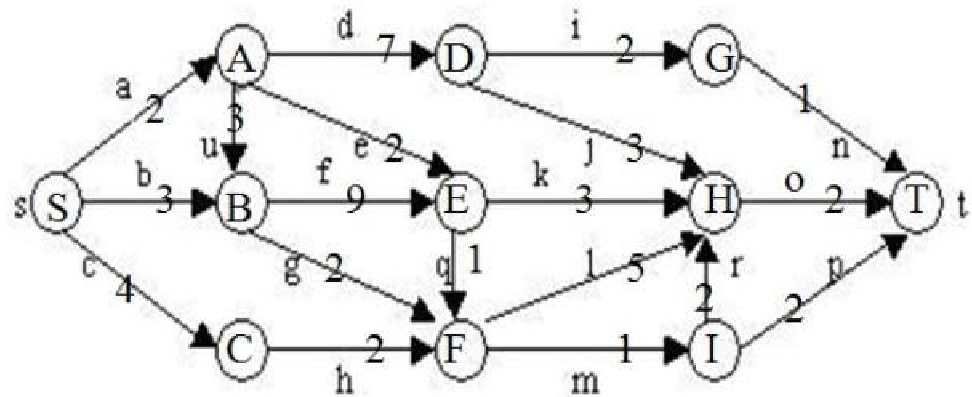


分支限界法案例4- 单源最短路径问题

图G

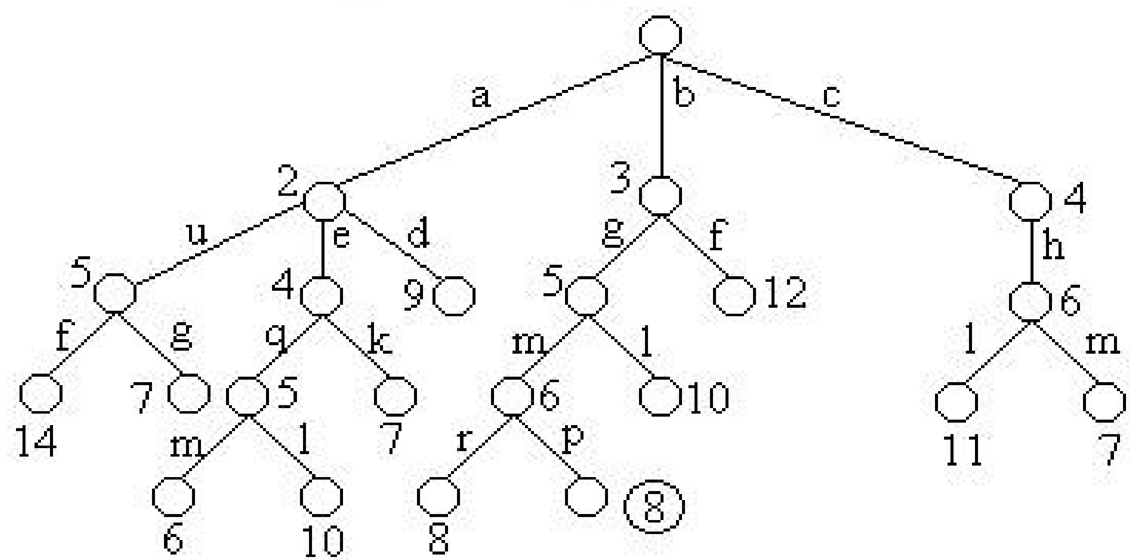
源点S

目标节点T



解空间树

路径长



优先队列式分支限界法-算法思想

- 从图G的源顶点s和空优先队列开始。结点s被扩展后，它的儿子结点被依次插入堆中。
- 从堆中取出具有最小当前路长的结点作为当前扩展结点，并依次检查与当前扩展结点相邻的所有顶点。
  - 如果从当前扩展结点i到顶点j有边可达，且从源出发，途经顶点i再到顶点j的所相应的路径的长度小于当前最优路径长度，则将该顶点作为活结点插入到活结点优先队列中。
- 继续扩展过程，直到活结点优先队列为空。

> 优先队列式分支限界法:用一极小堆来存储活结点表。其优先级是结点所对应的当前路长。

剪枝策略

- 在扩展结点的过程中，一旦发现一个结点的下界不小于当前找到的最短路长，则算法剪去以该结点为根的子树。
- 利用结点间的控制关系进行剪枝- 从源顶点s出发，2条不同路径到达图G的同一顶点。可将路长较长的路径所对应的子树剪去。

示例代码

```
while (true) {
    for (int j = 1; j <= n; j++)
     if ((c[E.i][j]<inf)&&(E.length+c[E.i][j]<dist[j])) {
      // 顶点i到顶点j可达，且满足控制约束
      dist[j]=E.length+c[E.i][j];
      prev[j]=E.i;
      // 加入活结点优先队列
      MinHeapNode<Type> N;
      N.i=j;
      N.length=dist[j];
      H.Insert(N);}
    try {H.DeleteMin(E);}        // 取下一扩展结点
    catch (OutOfBounds) {break;}  // 优先队列空
    }
}
```

顶点i和j间有边，且此路径长小于原先从原点到j的路径长

优化

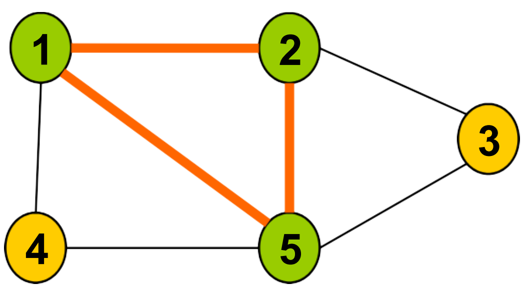
- 1. 能否用一个下界来代表优先级?

优化

- 1. 能否用一个下界来代表优先级?
- 2. 还有没有其他方法

分支限界法案例5- 最大团问题

- 团（clique）：社会团体，团体中的个体互相认 识。
- 最大团问题（Maximum Clique Problem, MCP）是图论中一个经典的组合优化问题，也是一类NP完全问题。



例：子集{1，2}是G的大小为2的完全子图。这个完全子图不是团，因为它被G的更大的完全子图{1，2，5}包含。{1，2，5}是G的最大团。{1，4，5}和{2，3，5}也是G的最大团。

回顾：回溯思路

- • 首先设最大团为一个空团，往其中加入一个顶点，然后依次考虑每个顶点，查看该顶点加入团之后是否仍然构成一个团，如果可以，考虑将该顶点加入团或者舍弃两种情况，如果不行，直接舍弃，然后递归判断下一顶点。
- • 判断当前顶点加入团之后是否仍是一个团：只需要考虑该顶点和团中顶点是否都有连接。
- • 剪枝策略：如果剩余未考虑的顶点数加上团中顶点数不大于当前解的顶点数，可停止继续深度搜索，否则继续深度递归当搜索到一个叶结点时，即可停止搜索，更新最优解和最优值。

上界函数及优先级

- cliqueSize表示与该结点相应的团的顶点数
- level表示结点在子集空间树中所处的层次
- cliqueSize+n-level+1作为顶点数上界upperSize的值
- upperSize实际上也是优先队列中元素的优先级
- 算法总是从活结点优先队列中抽取具有最大upperSize值的元素作为下一个扩展元素

算法思想：子集树

- 子集树的根结点是初始扩展结点，其cliqueSize的值为0
- 扩展内部结点时，
  - 先考察左子结点。将顶点 i 加入到当前团中，并检查可行性，即该顶点与当前团中其它顶点是否都有边相连。
    - 可行结点，将加入到子集树中并插入活结点优先队列。
  - 继续考察当前扩展结点的右子结点。
    - 当upperSize>bestn时，右子树中可能含有最优解，此时将右子结点加入到子集树中并插入到活结点优先队列中。

算法思想：while循环的终止条件

- 子集树中的一个叶结点(即n+1层结点)成为当前扩展结点。
  - 对于子集树中的叶结点，有upperSize = cliqueSize。
  - 此时活结点优先队列中剩余结点的upperSize值均不超过当前扩展结点的upperSize值，从而进一步搜索不可能得到更大的团，此时算法已找到一个最优解。