贪心算法

找零钱问题

- 问题描述：
  - 用面额为$d_1>d_2>\ldots>d_m$的最少数量的硬币找出金额为n的零钱。
  - 例：美国广泛使用的硬币面额为$d_1=25$，$d_2=10$，$d_3=5$，$d_4=1$。如何用这些面额给出48美分的找零？
  - 1个25美分，2个10美分，3个1美分
  - 基于“贪心”的想法可以将剩余的硬币数量降为最低

找零钱问题

- 例：假设有面值单位分别为1、4和6的硬币。如果需要找出金额为8的零钱。若采用贪心算法，得到的解决方案如何？
  - 先选取面值最大的，然后再选小面值的
  - 采用1枚面值为6的硬币，两枚面值为2的硬币
  - 更优的解：2枚面值为4的硬币
- 贪心算法并不能总是找出最优解
- 采用动态规划求最优解

贪心算法

- 贪心法建议通过一系列步骤来构造问题的解，每一步对目前构造的部分解做一个扩展，直到获得问题的完整解为止。
- 所做的每一步选择都必须满足以下条件：
  - 可行：即它必须满足问题的约束
  - 局部最优：它是当前步骤中所有可行选择中最佳的局部选择
  - 不可取消：选择一旦做出，在算法的后面步骤中就无法改变了

背包问题

- 问题描述
  - 给定n个体积分别为$s_1, s_2, \ldots, s_n$、价值分别为$v_1, v_2, \ldots, v_n$的物品和一个容量为C的背包，要找到非负实数$x_1, x_2, \ldots, x_n$，使和 $\sum_{i=1}^{n} x_i v_i$ 在约束 $\sum_{i=1}^{n} x_i s_i \le C$ 下最大。
  - 0/1背包问题：物品不可分
  - 一般背包问题：物品是可分割的
  - 在最优解中，$\sum_{i=1}^{n} x_i s_i = C$
  - 以某种合适的顺序选择每个物品，并尽可能将该物品装入背包。

背包问题

- 选择函数：
  - 选择剩余物品中价值最大的
  - 选择剩余物品中体积最小的
  - 选择单位体积上价值最高的
- 例：

s	10	20	30	40	50
v	20	30	66	40	60
v/s	2.0	1.5	2.2	1.0	1.2

选择	x_i	值
最大vi	0 0 1 0.5 1	146
最小si	1 1 1 1 0	156
最大vi/si	1 1 1 0 0.8	164

Algorithm **Greedy-KNAPSACK**

- **Input:** size vector s[1..n] and value vector v[1..n] of n items that are both sorted as non-increasing order according to the ratio v(i)/s(i); the capacity C of the knapsack, the total number of items n
- **Output:** the greedy optimal solution x[1..n]
- **for** *i* ← 0 **to** *n*
- x[*i*] ← 0
- **end for**
- cu ← C
- **for** *i* ← 0 **to** *n*
- **if** s[*i*] > cu **then**
- **exit**
- **end if**
- x(i)← 1
- cu ←cu-s(i)
- **end for**
- **if** i ≤n **then** x(i) ←cu/s(i) **end if**
- **return** x

活动安排问题

活动安排问题就是要在所给的活动集合中选出最大的相容活动子集合，是可以用贪心算法有效求解的很好例子

活动安排问题

- 1 问题描述：

设有n个活动的集合E={1,2,…,n}，其中每个活动都要求使用同一资源，如演讲会场等，而在同一时间内只有一个活动能使用这一资源。

每个活动i都有一个要求使用该资源的起始时间s_i和一个结束时间f_i,且$s_i<f_i$ 。如果选择了活动i，则它在半开时间区间$[s_i, f_i)$内占用资源。

若区间$[s_i, f_i)$与区间$[s_j, f_j)$不相交，则称活动i与活动j是相容的。也就是说，当$s_i \geq f_j$或$s_j \geq f_i$时，活动i与活动j相容。

活动安排问题就是在所给的集合中选出最大的相容活动子集合。

活动安排问题

```
int greedySelector(int s [ ], int n, int f [ ], bool a [ ])
   {  a[1]=true; j=1;
      count=1;
      for ( i=2;i<=n;i++) {
        if (s[i]>=f[j]) {
          a[i]=true;
          j=i;
          count++;
          }
        else a[i]=false;
        }
      return count;
   }
```

各活动的起始时间和结束时间存储于数组s和f中且按结束时间的非减序排列

活动安排问题

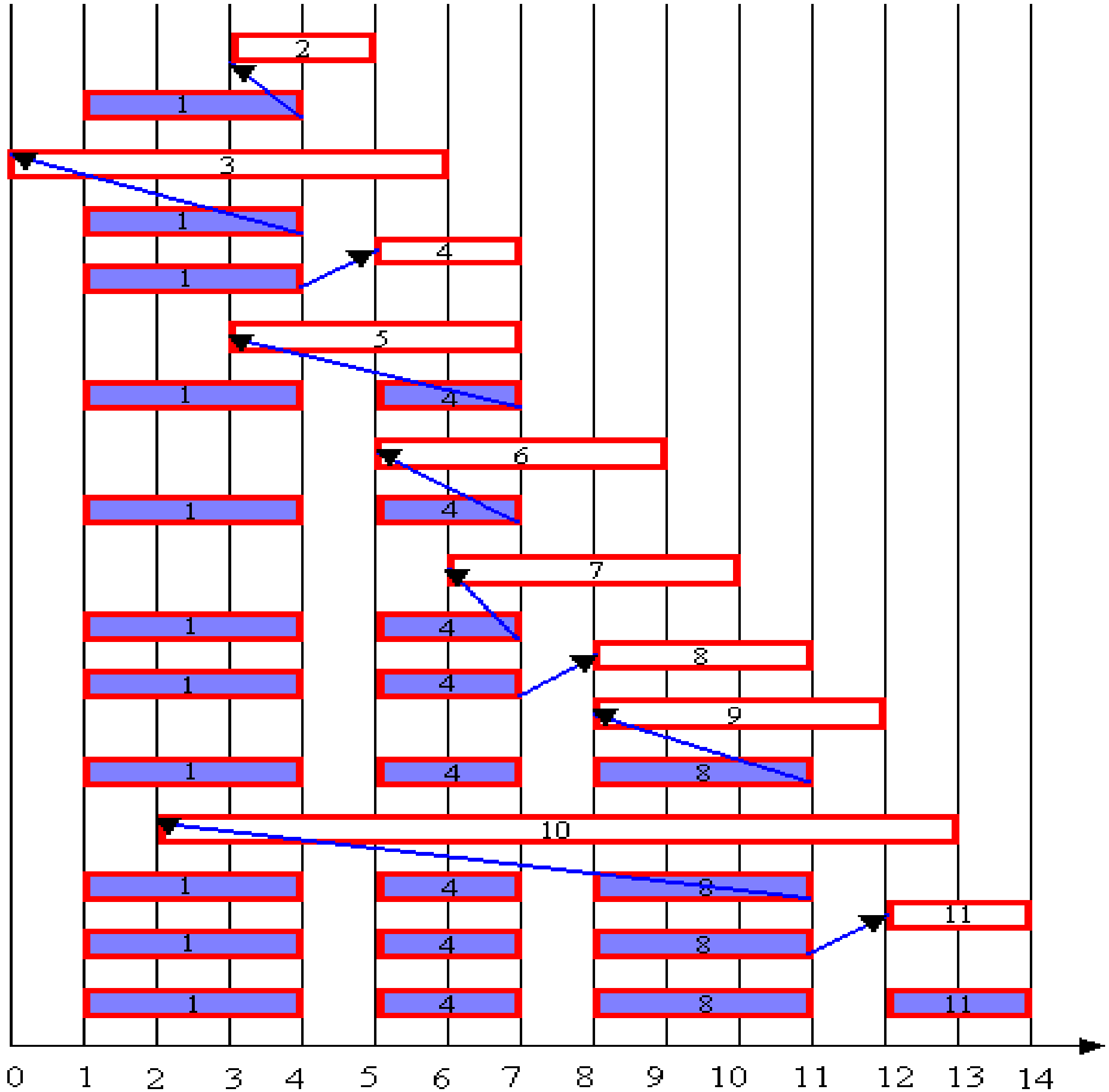
由于输入的活动以其完成时间的非减序排列，所以算法greedySelector每次总是选择具有最早完成时间的相容活动加入集合A中。直观上，按这种方法选择相容活动为未安排活动留下尽可能多的时间。也就是说，该算法的贪心选择的意义是使剩余的可安排时间段极大化，以便安排尽可能多的相容活动。

活动安排问题

例：设待安排的11个活动的开始时间和结束时间按结束时间的非减序排列如下：

i	1	2	3	4	5	6	7	8	9	10	11
S[i]	1	3	0	5	3	5	6	8	8	2	12
f[i]	4	5	6	7	8	9	10	11	12	13	14

活动安排问题



算法greedySelector的计算过程如左图所示。图中每行相应于算法的一次迭代。阴影长条表示的活动是已选入集合A的活动，而空白长条表示的活动是当前正在检查相容性的活动。

活动安排问题

- 正确性证明（用数学归纳法证明）

1 贪心选择性质

即证明活动安排问题总存在一个最优解从贪心选择开始。

4.2 活动安排问题

- 正确性证明（用数学归纳法证明）

1 贪心选择性质

设E={1,2,…,n}为所给的活动集合。由于E中的活动按结束时间的非递减排序，故活动1具有最早完成时间。首先证明活动安排问题有一个最优解以贪心选择开始，即该最优解中包含活动1.

4.2 活动安排问题

- 正确性证明（用数学归纳法证明）

1 贪心选择性质

设A⊆E是所给活动安排问题的一个最优解，且A中的活动也按结束时间非递减排序，A中的第一个活动是k。

活动安排问题

- 正确性证明（用数学归纳法证明）

  1 贪心选择性质

  若k＝1，则A就是以贪心选择开始的最优解。

  若k>1，设B=A-{k}∪{1}。因为f_1<=f_k且A中的活动是相容的。古B中的活动也是相容的。又由于B中的活动个数与A中的活动个数相同，故A是最优的，B也是最优的。即B是以选择活动1开始的最优活动安排。

  由此可见，总存在以贪心选择开始的最优活动安排方案。

活动安排问题

- 正确性证明（用数学归纳法证明）

2 最优子结构性质

在作出了贪心选择，即选择了活动1后，原问题简化为对E中所有与活动1相容的活动进行活动安排的子问题。即若A是原问题的最优解，则A’=A-{1}是活动安排问题的E’=$\{i \in E: S_i \geq f_1\}$的最优解。

活动安排问题

- 正确性证明（用数学归纳法证明）

2 最优子结构性质

反证法：若E’中存在另一个解B’，比A’有更多的活动，则将1加入B’中产生另一个解B，比A有更多的活动。与A的最优性矛盾。

活动安排问题

- 正确性证明（用数学归纳法证明）

  因此，每一步所作出的贪心选择都将问题简化为一个更小的与原问题具有相同形式的子问题。对贪心选择次数用归纳法可知，贪心算法greedySelector产生问题的最优解。

单源最短路径问题（Dijkstra算法）

- 问题描述
  - 已知有向图 $G = (V, E)$，每条边上的权值均为非负。其中一个结点s指定为源点，求从源点s到图中所有其它结点x的距离，即s到x的最短路径长度。
- Dijkstra算法—— 贪心策略
  - 按照最短路径长度递增的次序求得源点到达每一个顶点的最短路径

单源最短路径问题

- **最优子结构：**最短路径的**子路径是最短路径**
  - 对于一给定的带权有向图$G=(V,E)$，设$p=<v_1,v_2,\ldots,v_k>$是从v_1到v_k的最短路径。那么对于任意i,j，其中$1 \le i \le j \le k$，设$p_{ij}=<v_i,v_{i+1},\ldots,v_j>$为$p$中从顶点$v_i$到顶点$v_j$的子路径。那么，$p_{ij}$是从$v_i$到$v_j$的最短路径。

Dijkstra算法思想

- 假设V={$1,2,\ldots,n$}, 源点s=1。
- 初始顶点分为两个集合X = {1} 和Y = {2,3,...,n}。X集中的顶点是已求得最短路径的顶点；
- Y集中的每个顶点y都有一个标记$\lambda[y]$, 它是目前从源点到顶点y，中间只经过X集中的顶点的最短路径的长度；
- 每次从Y集中选择一个$\lambda[y]$值最小的顶点y加入X集；
- 一旦y加入X集，那么考察与y相邻的每个顶点w的标记，如果源点经过y到达w的路径更短，则更新该标记；
- 若$\delta[y]$表示从源点到v的距离，那么算法结束时，对每个顶点均有$\delta[y]= \lambda[y]$

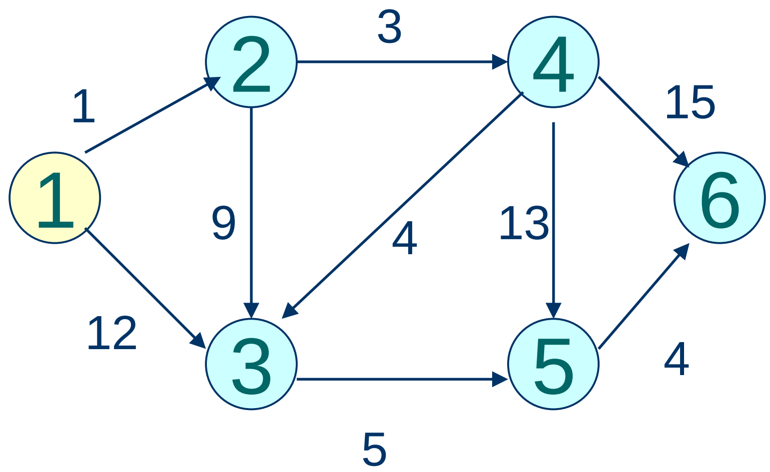
Algorithm 8.1 DIJKSTRA

Input: A weighted directed graph G=(V,E), where V={1,2,…,n}.

Output: The distance from vertex 1 to every other vertex in G.

1. $X=\{1\};\ Y \leftarrow V-\{1\};\ \lambda[1] \leftarrow 0$
2. **for y←2 to** *n*
3. **if** y is adjacent **to** 1 **then** $\lambda[y] \leftarrow length[1,y]$
4. **else** $\lambda[y] \leftarrow \infty$
5. **end if**
6. **end for**
7. **for** j←1 **to** *n-1*
8. *Let y ∈Y be such that λ[y] is minimum*
9. X←X ∪{y} {add vertex y to X}
10. Y←Y-{y} {delete vertex y form Y}
11. **for** each edge (y,w)
12. **if** w ∈Y **and** λ[y]+length[y,w]< λ[w] **then**
13. λ[w]← λ[y]+length[y,w]
14. **end for**
15. **end for**

Dijkstra算法



Y

2	3	4	5	6
1	12	∞	∞	∞

3	4	5	6
10	4	∞	∞

3	5	6
8	17	19

5	6
13	19

6
17

X

{1}

{1,2}

{1,2,4}

{1,2,4,3}

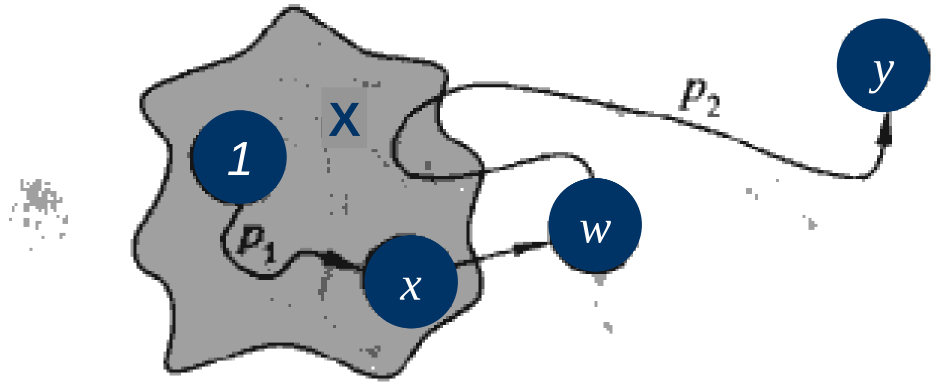
{1,2,4,3,5}

{1,2,4,3,5,6}

Dijkstra算法总是在Y集中选择离源点最近的顶点加入集合X，故称该算法使用了贪心策略。

Dijkstra算法的贪心选择性质（正确性证明）

- 贪心策略并非总是能获得全局意义上的最理想结果，但Dijkstra算法确实得到了最短路径。关键要证明，当顶点y加入集合X时，有$\delta[y]=\lambda[y]$



数学归纳法证明

Dijkstra算法

- 定理8.1：给出一个边具有非负权值的有向图G和源点s， Dijkstra算法在时间$\Theta(n^2)$ 内找出从s到其他每一顶点距离的长度。
- 对Dijkstra算法的改进
  - 针对稠密图
  - 利用小顶堆数据结构，在O(log)时间内从Y集中选出顶点y
  - 采用邻接表为图的存储结构

Algorithm 8.2 SHORTESTPATH

Input: A weighted directed graph G=(V,E), where V={1,2,..,n}.

Output: The distance from vertex 1 to every other vertex in G.

Assume that we have an empty heap H at the beginning.

1.Y←V-{1}; λ[1]←0;   key(1)← λ[1]
2.**for** y←2 **to** n
3. **if** y is adjacent to 1 **then**
4. λ[y]=length[1,y]
5. key(y)← λ[y]
6. INSERT(H,y)
7. **else**
8. λ[y]← ∞
9. key[y]← λ[y]
10. **end if**
11. **end for**

NEXT PAGE

Algorithm 8.2 SHORTESTPATH

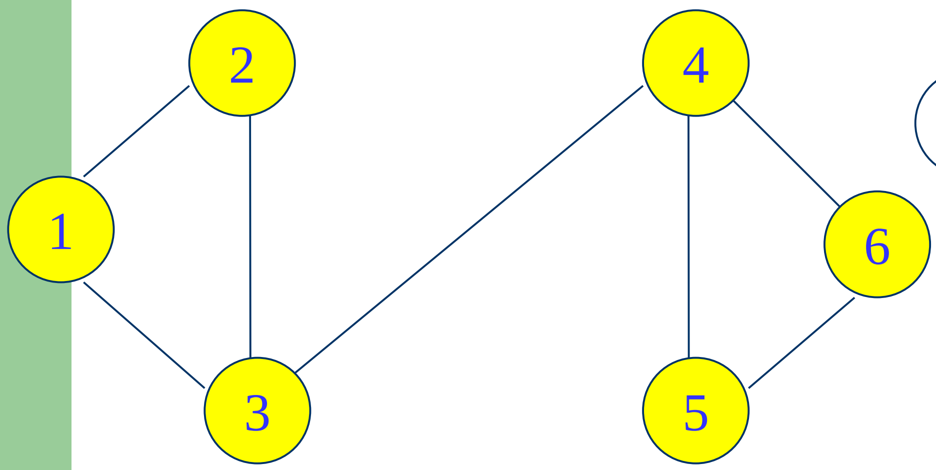
```
for j←1 to n-1
    y←DELETEMIN(H)
    Y←Y-{y}
    for each vertex w ∈Y that is adjacent to y
        if λ[y]+length[y,w]< λ[w] then
            λ[w]← λ[y]+length[y,w]
            key(w)← λ[w]
        end if
        if w   H then INSERT(H,w)
        else SIFTUP(H,H^-1(W))
        end if
    end for
end for
```

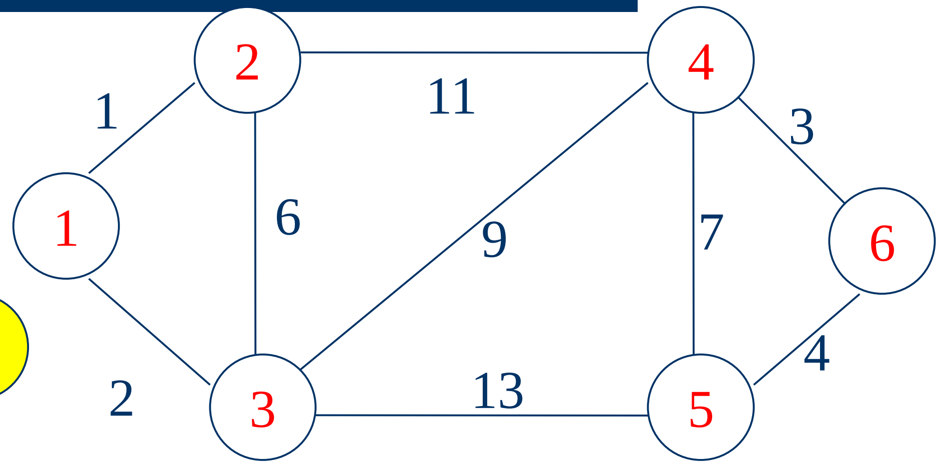
O(mlog n)

最小生成树问题（Kruskal算法）

- 定义8.1：设G = (V, E)是一个具有含权边的连通无向图。G的一棵生成树(V,T)是G的作为树的子图。如给G加权并且T的各边的权的和为最小值，那么(V,T)就称为最小耗费生成树，简称生成树。
- Kruskal算法：
  - 对G的边以非降序权重排列；
  - 对排好序的每条边，如果将其加入T不会形成回路，则加入树T中；否则将其丢弃。

例： Kruskal算法





(1,2)	(1,3)	(4,6)	(5,6)	(2,3)	(4,5)	(3,4)	(2,4)	(3,5)
1	2	3	4	6	7	9	11	13

若构成回路，则丢弃该边

Kruskal算法的实现

- 表示森林的数据结构
  - 用不相交集数据结构来表示森林
  - 初始时，图的每个顶点由包含一个顶点的树表示
  - 算法执行时，森林中的每个连通分量由一棵树来表示
  - 选择权值最小的边将两个连通分量连接起来由Union操作实现

Algorithm 8.3 KRUSKAL

Input: A weighted connected undirected graph G=(V,E) with n vertices.

Output: The set of edges T of a minimum cost spanning tree for G.

1.Sort the edges in E by nondecreasing weight
2.**for** each vertex $v \in V$
3. MAKESET{v}
4.**end for**
5.T={}
6.**while** |T|<n-1
7. Let (x,y) be the next edge in E.
8. **if** FIND(x) $\neq$FIND(y) **then**
9. Add(x,y) to T
10 UNION(x,y)
11 **end if**
12. **end while**

Kruskal算法正确性证明

- 引理8.2：在含权无向图中，Kruskal算法能正确地找出最小生成树
- 证明：
  - 对T的大小实施归纳法，证明T是最小生成树边集的子集
  - 初始时，T={}，命题成立；
  - 设加入边e = (x,y)之前命题成立，即有$T \subset T^*$，其中T^*是最小生成树边的集合；
  - 设X是包含x的子树的顶点集，$T' = T \cup \{e\}$，需证明T'也是最小生成树边集T^*的子集。

Kruskal算法正确性证明

- 由于T⊂T*，若T*包含e，则T'=T∪{e} ⊂T*；
- 若T*不包含e，则T*∪{e}必定包含以e为一条边的回路；
- e=(x,y)连接了X中的一个顶点和V-X中的另一个顶点，则T*中必定存在另一条边e'=(w,z)，其中w∈X，z∈V-X；
- 由于e在e'之前添加，故cost(e') ≥cost(e)；
- 构造T**=T*-{e'}∪{e}，则T'⊂T**，且T**是最小生成树边的集合。

Kruskal算法时间效率分析

- 第1和第2步分别花费 O(mlogm) 和 Θ(n), m = |E|;
- 第7步执行n-1次，需要花费O(n);
- 第9步花费 Θ(1), 最多执行m次,总共花费 O(m);
- Union操作执行n-1 次，而find 操作最多执行 2m 次.，两个操作总的花费为O(mlog*n);
- 算法总的运行时间取决于排序算法，即O(mlogm);
- 定理8.3：在一个有m条边的含权无向图中，Kruskal算法可在O(mlogm)时间内找出最小生成树

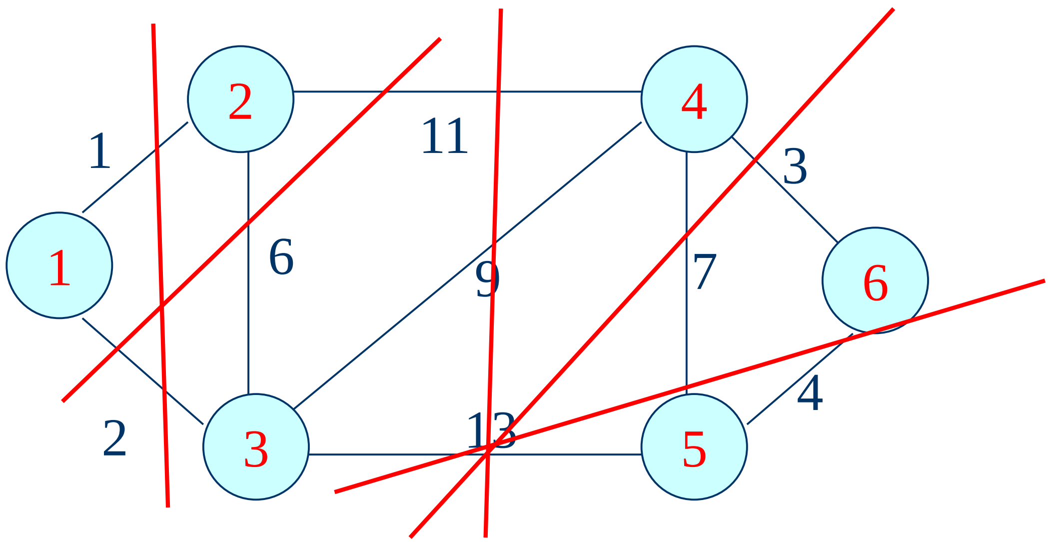
最小生成树问题（Prim算法）

- 设G=(V,E), V取整数集合{1,2,…,n}.
- 算法从建立两个顶点集开始: X={1}和Y={2,…,n}, 然后生长生成树，每次生成一条边。
- 生成的边(x,y)具有以下性质： x∈X, y ∈Y， 且该边是所有满足上述条件的边中权值最小者。
- 然后将y从Y集移到X集。
- 重复生成边的步骤直到Y集为空为止。

Prim算法

1. $T \leftarrow \{\}$; $X \leftarrow \{1\}$; $Y \leftarrow V-\{1\}$
2. **while** $Y \neq \{\}$
3. Let (x,y) be of minimum weight such that $x \in X$ and $y \in Y$
4. $X \leftarrow X \cup \{y\}$
5. $Y \leftarrow Y-\{y\}$
6. $T \leftarrow T \cup \{(x,y)\}$
7. **end while**

例：Prim算法生成生成树



Algorithm 8.4 PRIM

Input: A weighted connected undirected graph G=(V,E), where V={1,2,..,n}.

Output: The set of edges T of a minimum cost spanning tree for G.

1.T←{}; X←{1}; Y←V-{1}

2.**for** y←2 **to** n

3. **if** y adjacent to 1 **then**

4. N[y]←1

5. C[y]←c[1,y]

6. **else** C[y]←∞

7. **end if**

8.**end for**

NEXT PAGE

Algorithm 8.4 PRIM

```
for j←1 to n-1   {find n-1 edges}
    Let y∈Y be such that C[y] is minimum
    T←T∪{y,N[y]}          {add edge (y,N[y]) to T}
    X←X∪{y}               {add vertex y to X}
    Y←Y-{y}               {delete vertex y from Y}
    for each vertex w ∈Y that is adjacent to y
        if   c[y,w]<C[w]    then
            N[w]←y
            C[w]←c[y,w]
        end if
    end for
end for
```

Prim算法的正确性证明

- 引理8.3：在含权无向图中，Prim算法能正确地找出最小生成树
- 证明：
  - 对T的大小实施归纳法，证明(X,T)是最小生成树的子树
  - 初始时，T={}，命题成立；
  - 假设添加边e=(x,y)之前命题成立， 其中x∈X, y∈Y
  - 设X’=X∪{y}，T’=T∪{e}，需证明G’=(X’,T’)也是最小生成树的子树（参见P155）

Prim算法的效率分析

- 第1步花费Θ(n);
- 第2步循环花费Θ(n)时间.第3-6步花费O(n);
- 第10步搜索离X最近的顶点y，每次迭代花费时间Θ(n). 搜索需执行 n-1次，故第10步总体花费 $\Theta(n^2)$;
- 第14步共执行 2m 次，第15步测试执行m次，第16、17步最多执行m次；
- 因此，算法的时间复杂度为 $\Theta(m+n^2)$.

Prim算法的改进

- 利用最小堆数据结构，使得可以在O(logn)时间内取得Y集中的顶点y，这个y和V-Y集中一个顶点连接的边的代价是最小的。

改进的Prim算法

Input: A weighted connected undirected graph G=(V,E), where V={1,2,..,n}.

Output: The set of edges T of a minimum cost spanning tree for G.

Assume that we have an empty heap H at the beginning.

1. T←{}; Y←V-{1}
2. **for** y←2 **to** n
3. **if** y is adjacent to 1 **then**
4. N[y]←1
5. key(y)←c[1,y]
6. INSERT(H,y)
7. **else** key(y)←∞
8. **end if**
9. **end for**

NEXT PAGE

改进的Prim算法

```
for j←1 to n-1                          {find n-1 edges}
    y←DELETEMIN(H)
    T←T∪{(y,N|y|)}                      {add edge (y,N|y|) to T}
    Y←Y-{y}                             {delete vertex y from Y}
    for each vertex w∈ Y that is adjacent to y
        if c[y,w]<key(w) then
            N[w]←y
            key(w)←c[y,w]
        end if
        if w    H then INSERT(H,w)
        else SIFTUP(H,H^-1(w))
    end for
end for
```

文件压缩问题（Huffman树算法）

- 设有100,000个字符的文件，其中只包含有 a, e, t, r, f, d 六种不同的字符，且字符出现的频度不尽相同。计算机中能够表达的信息只有0和1。怎样编码才能使文件的编码长度最短？

文件编码

1 1 0 0 1 1 0 0　　1 1 0 0 1 1 0 0

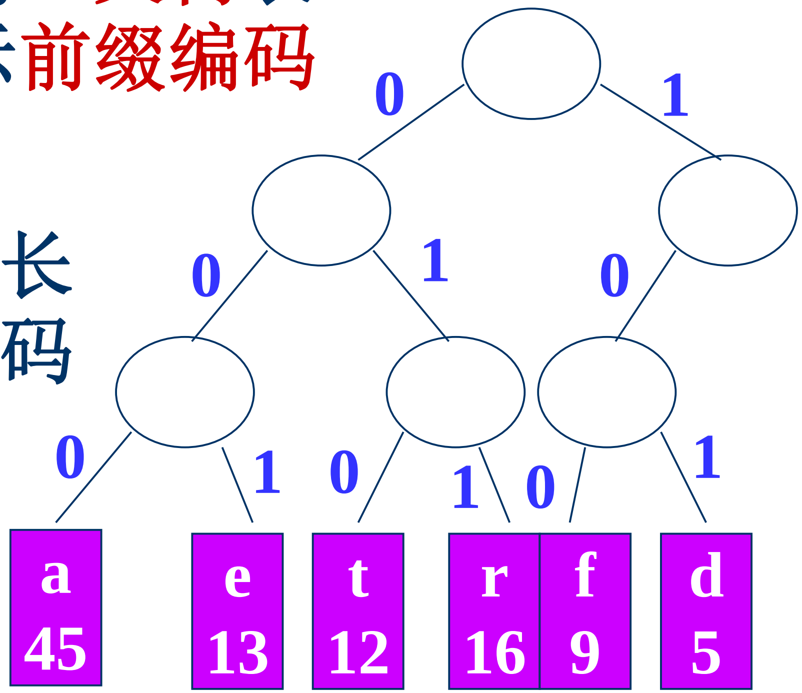
	a	e	t	r	f	d
频度(千字)	45	13	12	16	9	5
等长编码	000	001	010	011	100	101
变长编码1	0	001	010	110	1101	1100
变长编码2	0	101	100	111	1101	1100

前缀编码

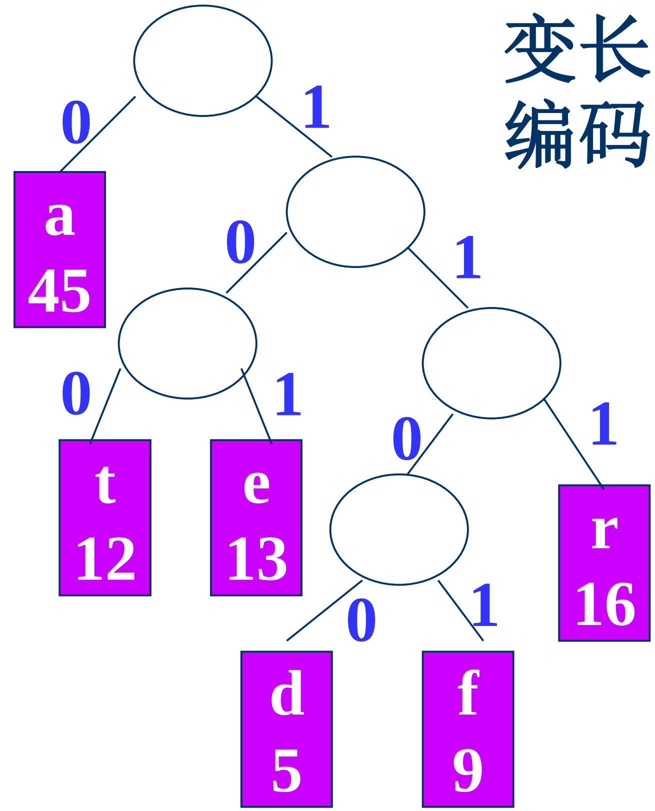
前缀编码

用二叉树表示前缀编码

等长编码



变长编码



文件编码长度 $B(T)=\sum_{c\in C} f(c)d_T(c)$

Huffman算法

- 贪心策略：将短的编码分配给高频字符，长的编码分配给低频字符
  - 构造 n 棵**只含根**结点的二叉树
  - 选择两个**权值最小**的二叉树，构造新的二叉树
  - **重复**该过程，直到**只剩一棵**二叉树为止

Huffman算法的正确性

- 具有贪心选择性质
  - 设x和y是字母表C中具有最低频度的两个字符，则存在C的一种最优前缀编码，其中x和y的编码长度相同但最后一位不同。
  - 通过合并构造出的树的总代价是各次合并的代价和，而在每一步所有可能的合并中，*Huffman*选择一个代价最小的合并。

Huffman算法的正确性

- 具有最优子结构性质
  - 设C为给定字母表，x和y是C中具有最低频度的两个字母。C’为字母表移去x和y，再加上新字符z后的字母表，即C’=C-{x,y} ∪{z}。z的频度为x和y的频度之和。
  - 设T’为表示字母表C’上的最优前缀编码的任意树。那么，将T’的叶子结点z替换成具有x和y为孩子的内部结点所得到的树T，表示字母表C上的一个最优前缀编码。

Algorithm 8.6 HUFFMAN

Input: A set $C=\{c_1,c_2,\ldots,c_n\}$ of n characters and their frequencies $\{f(c_1),f(c_2),\ldots,f(c_n)\}$.

Output: A Huffman tree (V,T) for C.

1.Insert all characters into a min-heap H according to their frequencies.

2. V←C; T={}

3. **for** j←1 **to** n-1

4. *c←DELETEMIN(H)*

5. *c'←DELETEMIN(H)*

6. *f(v)←f(c)+f(c')* {*v is a new node*}

7. *INSERT(H,v)*

8. *V=V U{v}* {*Add v to V*}

9. *T=T U{(v,c),(v,c')}* {Make c and c' children of v in T}

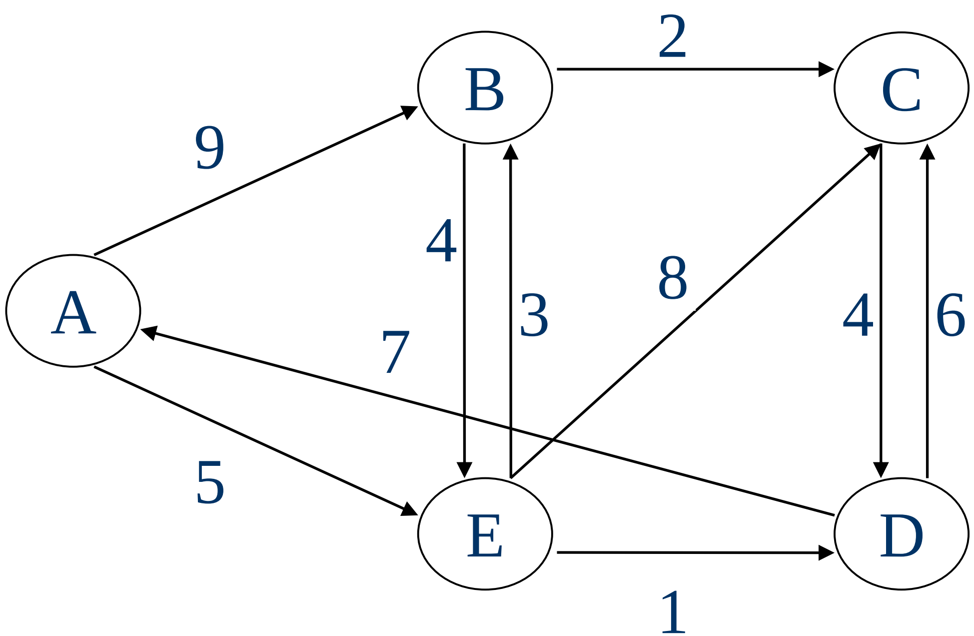
10. **end for**

O(nlogn)

思考题

- 设有n个客户同时等待一项服务，其中客户i需要的服务时间为t_i（$1<=t_i<=n$），应如何安排n个客户的服务次序能使得平均等待时间达到最小？其中，客户的等待服务时间为客户从开始等待到最后服务完成之间的时间，平均等待时间是n个客户等待服务时间的总和除以n。请证明你的策略的正确性 。

- 给定一个带权有向图G2=(V,E)，且此图上的权值非负，如图2所示。请详细写出利用Dijkstra算法求解从源点A到达其余各点的最短路径的详细过程，并报告每条最短路径及其长度。



- 已知有7件物品, 其重量分别为2,3,5,7,1,4,1，其对应的价值分别为10,5,15,7,6,18,3，背包允许的最大载重量为15，物品只能选择全部装入或不装入背包。请用贪心策略和动态规划法分别对该问题进行求解，并讨论贪心算法是否可以得到该背包问题的最优解。