计算理论与
算法分析设计

教材:

[1][王] 王晓东,计算机算法设计与分析(第4版),电子工业.

[2][S] 唐常杰等译, Sipser著, 计算理论导引, 机械工业.

参考资料:

[3][C] 潘金贵等译, Cormen等著, 算法导论, 机械工业.

[4][M] 黄林鹏等译, Manber著, 算法引论-一种创造性方法, 电子.

[5][刘] 刘汝佳等, 算法艺术与信息学竞赛, 清华大学.

[6][L] Lewis等著, 计算理论基础, 清华大学.

第四章 贪心算法
Greedy

[王]本章内容

4.1 活动安排问题

4.2 贪心算法的基本要素

4.3 最优装载 见上一章课件

4.4 哈夫曼编码

4.5 单源最短路 见上一章课件

4.6 最小生成树

4.7 多机调度问题证明

第四章 贪心算法
Greedy

!!贪心不一定正确(0-1背包), 需要证明

1. 活动安排问题
2. 贪心算法基本要素
3. Huffman算法
4. 最小生成树
5. 多机调度问题

活动安排问题

n个活动申请一个活动室, 各活动起始终止区间(s_i, f_i)

- 输入: n, (s_i, f_i), i = 1:n
- 输出: 最大相容活动子集(无冲突, 活动个数最多)

举例: {(1,4), (3,5), (0,6), (5,7)}　　//解不唯一

- 最优解有什么特点? 贪心, OSP(A[i]=true或false)
- 最优解可以包含 最早结束的 或 最晚开始的 //举例

先给出算法, 再证明正确性

1. 按终止时间排序 $f_1 \le f_2 \le \ldots \le f_n$. A[1] = true; pt = 1;
2. 对 i = 2:n
3. 　若 s[i] >= f[pt], 则 A[i] = true, pt = i,
4. 　　否则 A[i] = false

活动安排算法正确性证明

n个活动申请一个活动室, 活动起始终止区间(s_i, f_i)

- 输入: n, (s_i, f_i), $i = 1{:}n$, 输出: 最大相容活动子集
- 第一步 证明贪心选择性质:

  存在最优解包含相容最早结束的活动1
- 第二步 证明最优子结构性质:

  若A是包含活动1的最大相容活动集(最优策略),

  则A-{1}是区间$[f_1, f_n]$上的最大相容活动集
- 由此用数学归纳法, 就可以证明算法正确性

第四章 贪心算法
Greedy

!!贪心不一定正确(0-1背包), 需要证明

1. 活动安排问题
2. 贪心算法基本要素
3. Huffman算法
4. 最小生成树
5. 多机调度问题

贪心算法的基本要素

对于一个具体问题, 怎么知道能否用贪心算法

- 贪心选择性质和最优子结构性质(比较矩阵连乘)
- 贪心选择性质

  对比: 矩阵连乘, 0-1背包, 分数背包

  贪心算法第一基本要素, 与DP主要区别

  自顶向下计算
- OSP: 最优策略的子策略也是最优 //动规, 贪心
- 正确性证明一般过程:

  贪心选择+OSP+数学归纳法

  条件: 子问题与原问题类似, 相对独立

  子问题的最优解和贪心选择联合得整体最优解

第四章 贪心算法
Greedy

!!贪心不一定正确(0-1背包), 需要证明

1. 活动安排问题
2. 贪心算法基本要素
3. Huffman算法
4. 最小生成树
5. 多机调度问题

哈夫曼编码

- **Huffman编码广泛应用于数据文件压缩**
- **压缩率通常在20%~90%之间**
- **对文章先统计各字母出现频率, 再选择最优编码**
- **输入: 字母集和各字母出现频率**
- **输出: 各字母的最优编码(压缩率最大)**
- **前缀码: 任何代码都不是其它代码的前缀**

  例 a:0, b:101, c:100, d:11

  可以唯一解码011101100 = adbc

  定长码是前缀码: a:00, b:01, c:10, d:11

不同编码举例

	a	b	c	d	e	f
频率(千次)	45	13	12	16	9	5
定长码	000	001	010	011	100	101
变长码	0	101	100	111	1101	1100

使用定长码编码的存储空间:

3×(45+13+12+16+9+5) = 300kb

使用变长码编码的存储空间:

1×45+3×(13+12+16)+4×(9+5) = 224kb

节约空间约25%. 构造编码的原则?

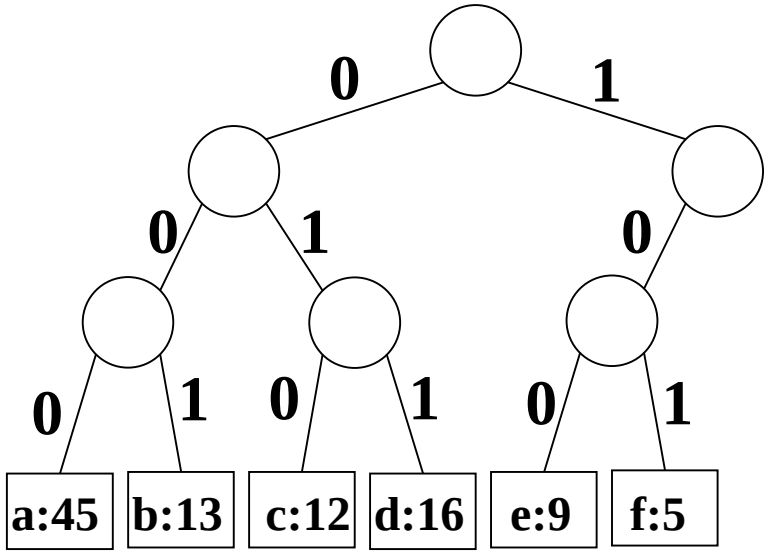
Huffman编码是最优前缀码

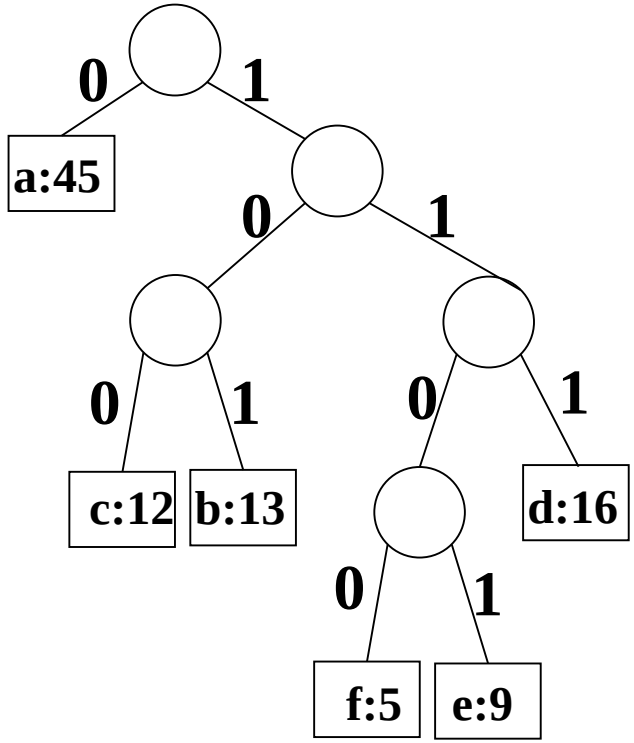
编码的数据结构

	a	b	c	d	e	f	存储
频率(千次)	45	13	12	16	9	5	100
定长码	000	001	010	011	100	101	300kb
变长码	0	101	100	111	1101	1100	224kb

选择数据结构

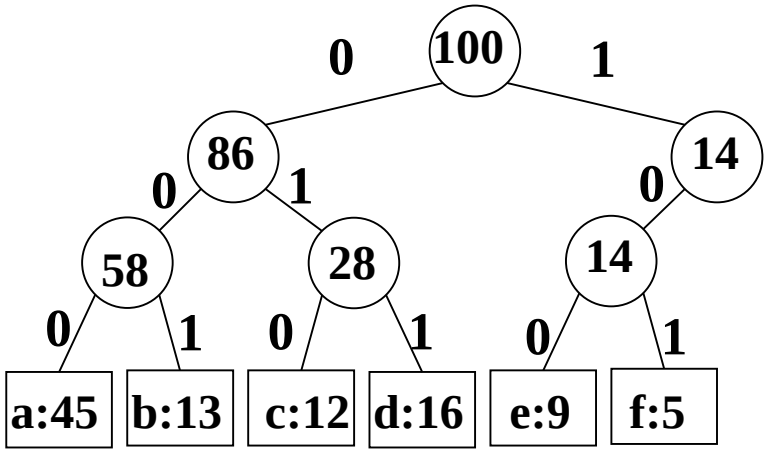
二叉树:

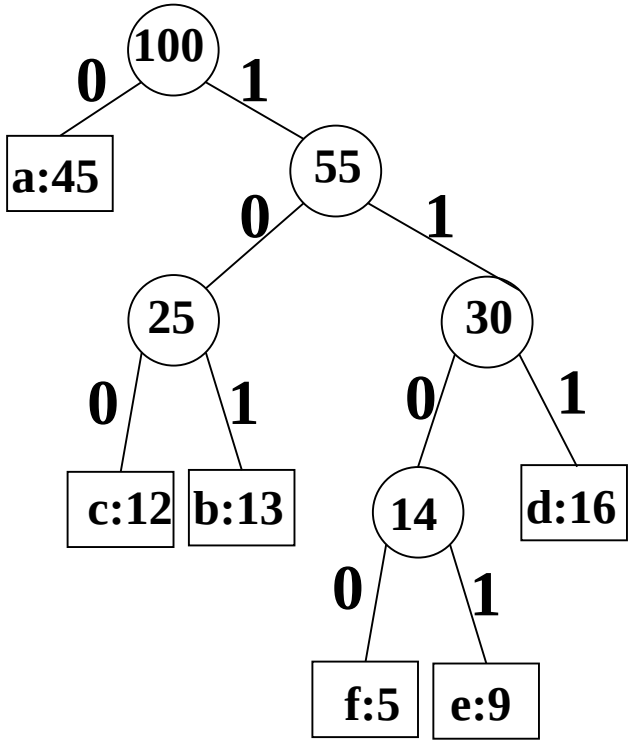




好编码的特点

	a	b	c	d	e	f	存储
频率(千次)	45	13	12	16	9	5	100
定长码	000	001	010	011	100	101	300kb
变长码	0	101	100	111	1101	1100	224kb



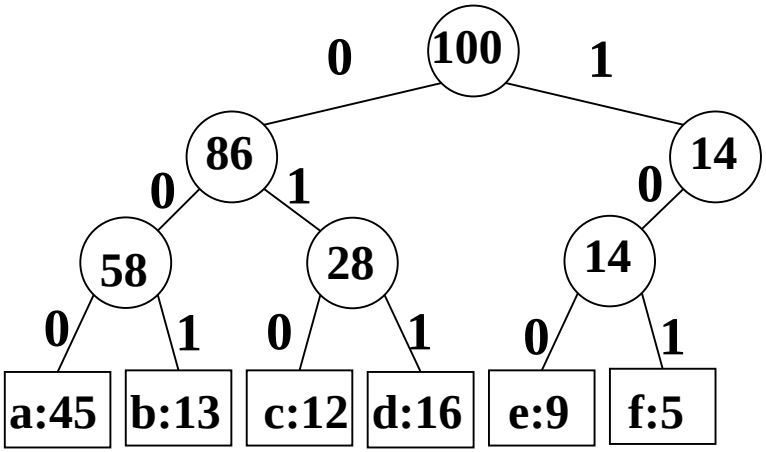


- 大频率编码短
- 小频率编码长

- 正则二叉树
- 相对平衡

构造优化模型

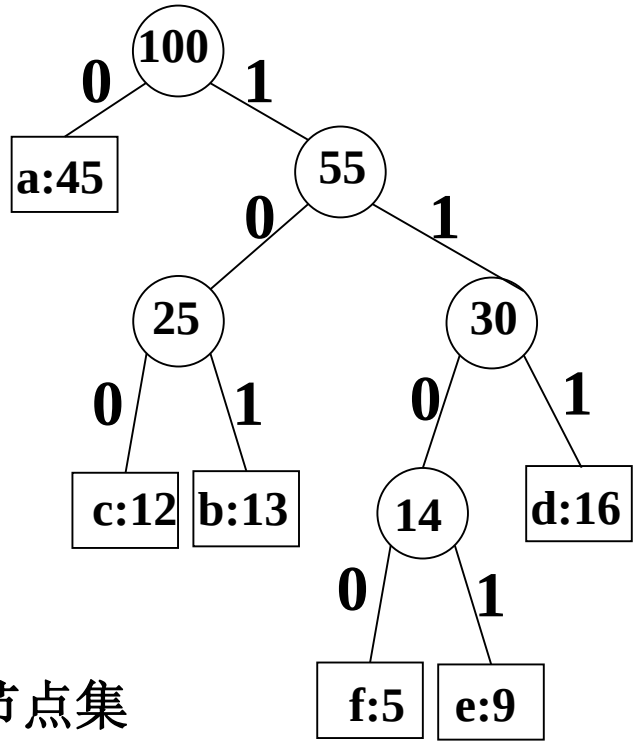
	a	b	c	d	e	f	存储
频率(千次)	45	13	12	16	9	5	100
定长码	000	001	010	011	100	101	300kb
变长码	0	101	100	111	1101	1100	224kb



叶节点深度

=

编码长度



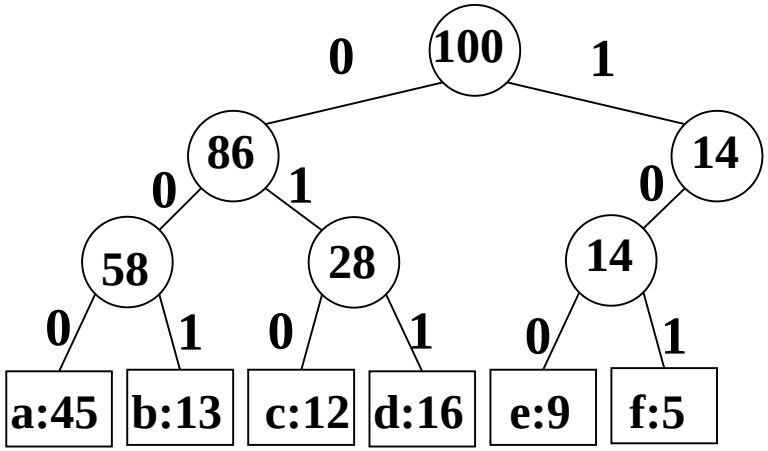
优化目标: 叶节点频率乘深度的和最小

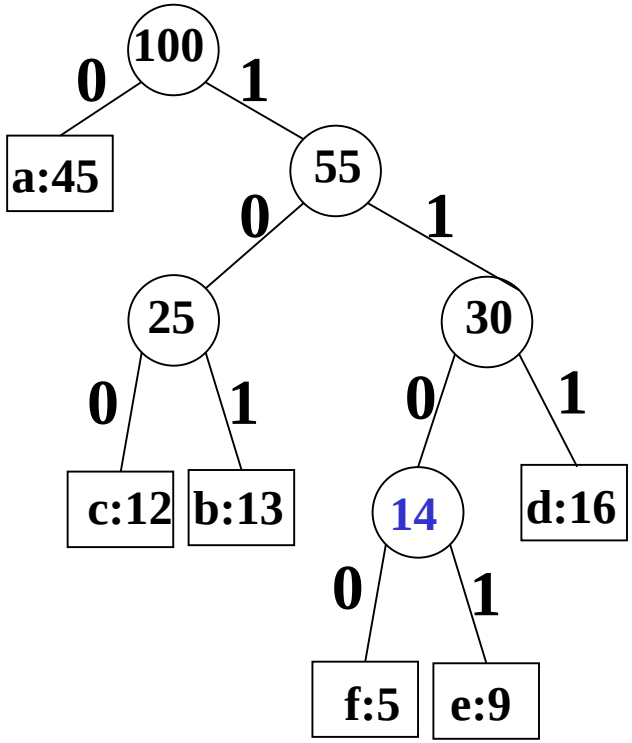
$$\min_{T} \sum_{c \in C} f(c) d_T(c)$$

其中 f(c) 是c的频率,

$d_T(c)$ 是c的深度, C是叶节点集

观察规律



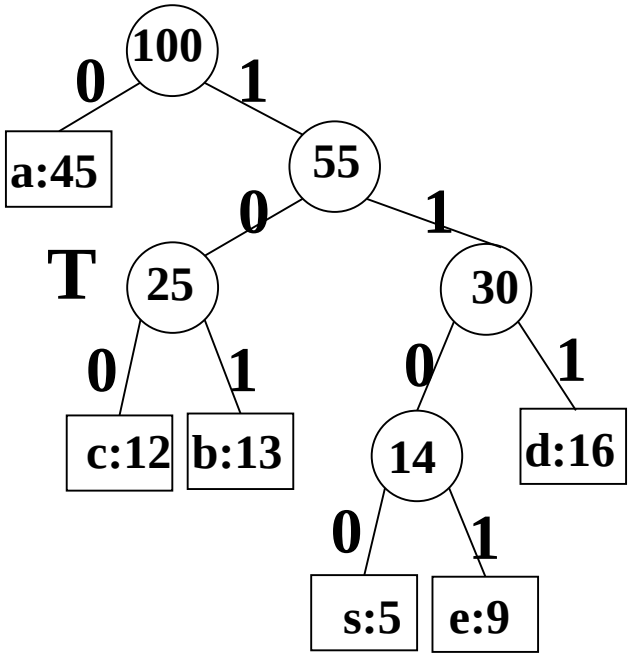


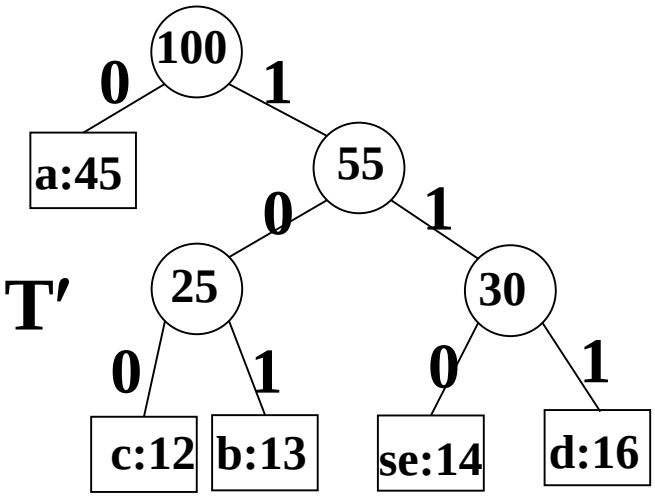
$$\min_{T} \sum_{c \in C} f(c) d_T(c)$$

- 正则二叉树
- 相对平衡
- f(c)最小?

- f(c)最小两符号深度最大(贪心选择性质)
- 接下来怎么做? 构造子结构? **OSP**
- 合并fe得新二叉树(子结构)

最优编码T的性质





$$B(T)=\sum_{c\in C} f(c)d_T(c)$$

- 正则二叉树
- 贪心选择性质:

  若f(c)最小, 则$d_T(c)$最大

  f(c)最小的两个是兄弟(s,e)
- OSP:

  构造T的子结构T′

  合并T中f(c)最小的两个得T′

  B(T)=B(T′)+f(s)+f(e)?

  若T最优, 则T′最优

Huffman算法

输入C[1:n], f[1:n] //字母集C和频率f

1. Q=C //建优先队列Q，O(n)

2. 对 i = 1:n-1 //循环n-1次

3. 分配新节点z

4. 取出Q中f最小的x作为z的左孩子 //O(log n)

5. 取出Q中f最小的y作为z的右孩子 //O(log n)

6. f[z] = f[x] + f[y]

7. 将z插入Q中 //O(log n)

8. 取出Q中节点作为树根 //?

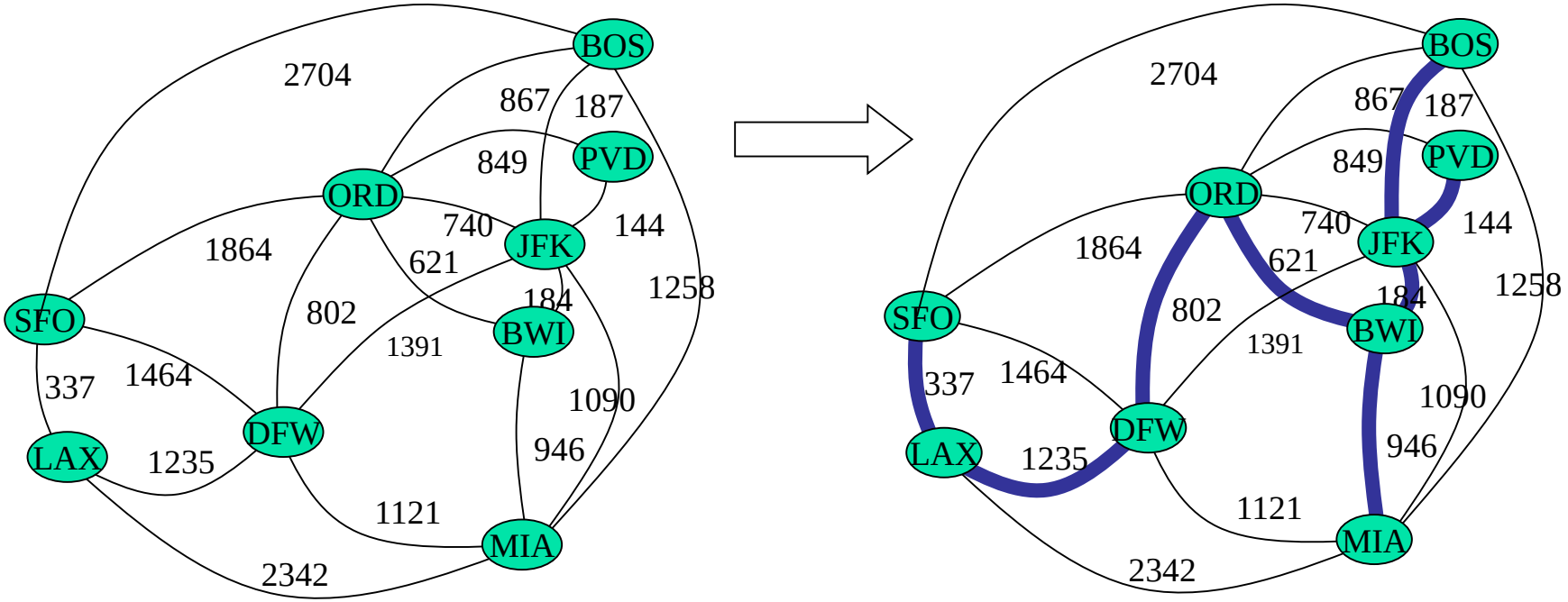
第四章 贪心算法
Greedy

!!贪心不一定正确(0-1背包), 需要证明

1. 活动安排问题
2. 贪心算法基本要素
3. Huffman算法
4. 最小生成树
5. 多机调度问题

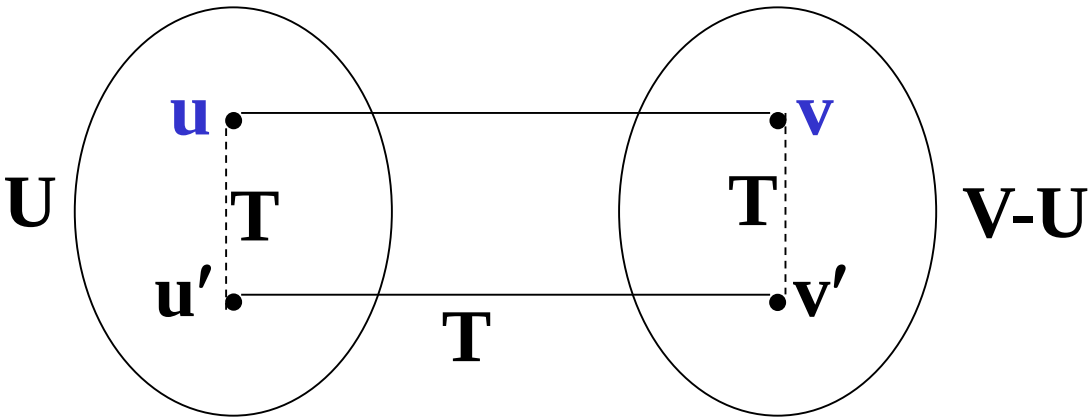
最小生成树(minimum spanning tree)

- 无向连通带权图G=(V,E,w).
- G的生成树是G的包含所有顶点的一颗子树
- 若G的生成树T’在所有G的生成树中各边权总和最小，则T’称为G的最小生成树(MST)



MST性质(贪心选择+归纳)

- 定理: 无向连通带权图G=(V,E,w). U⊂V, A⊂E.
  若A是某MST的子集, A没有边连接U和V-U,
  且(u,v)是G中连接U和V-U的权最小的边,
  则G存在MST包含A和边(u,v).
- 证明: 如图, 设T是包含A的MST, (u,v)∉T, (u′,v′)∈T.
  则必有 w(u,v) ≤ w(u′,v′). 由T去(u′,v′)添(u,v)得树T′.



T′也是MST
包含A和(u,v)

MST算法正确性证明

- 输入: G=(V,E,w)无向连通; 输出: G的MST
- MST性质: U⊂V, A⊂E. 若A是某MST的子集, A∩(U,V-U)=∅, (u,v)在(U,V-U)中权最小, 则存在MST包含A和(u,v).
- Prim: 维护U⊆V, 初始任取V中一点

1. 当U!=V, 取U到V-U的最小权边(u,v), 将v加入U

正确性证明: U, A是已有边集

- Kruskal: 维护A,Q⊆E, 初始A为空集, Q=E升序排列

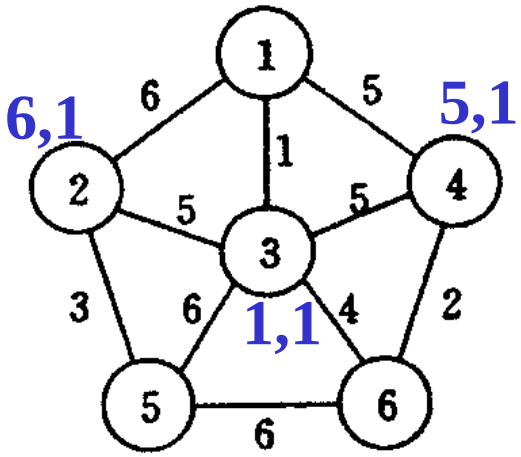
1. 当Q非空, 取Q最小权边(u,v),
2. 若(u,v)加入A会形成回路, 舍去; 否则加入A.

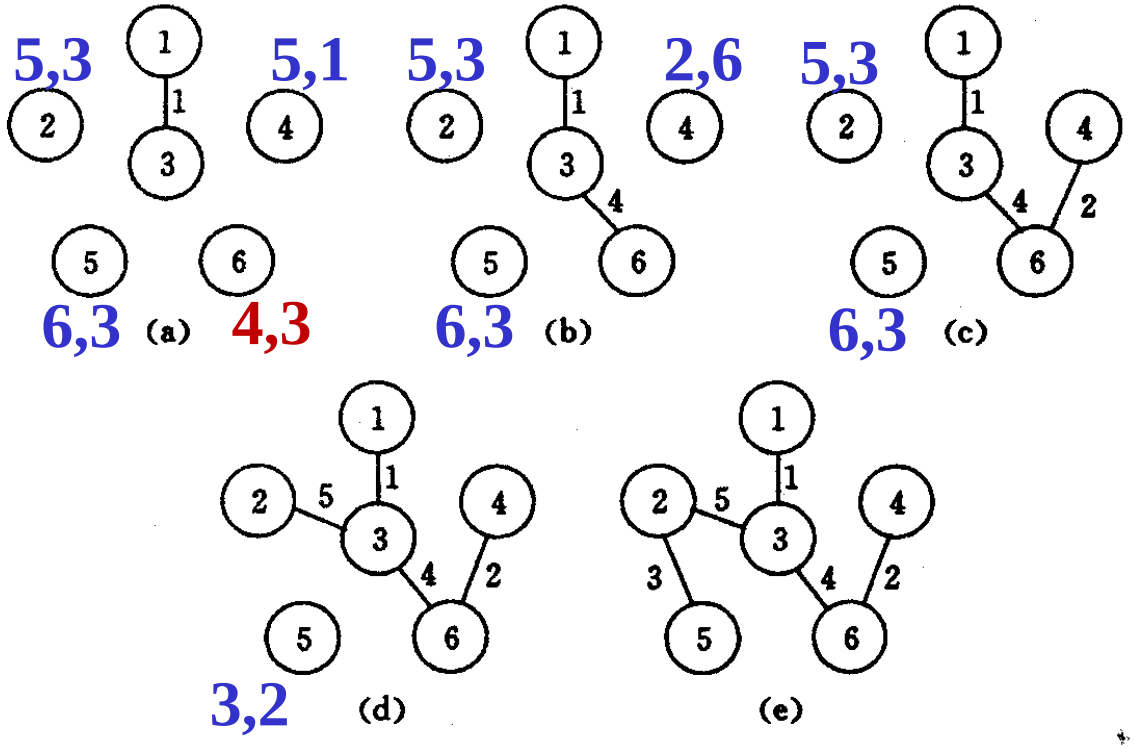
正确性证明: A, U={已经挑出边中与u连通的节点}

Prim算法

- 输入: G=(V,E,w)无向连通; 输出: G的MST
- 维护U⊆V, 初始任取V中一点

当U!=V, 取U到V-U的最小权边(u,v), 将v加入U.





Prim算法O(|V|²)

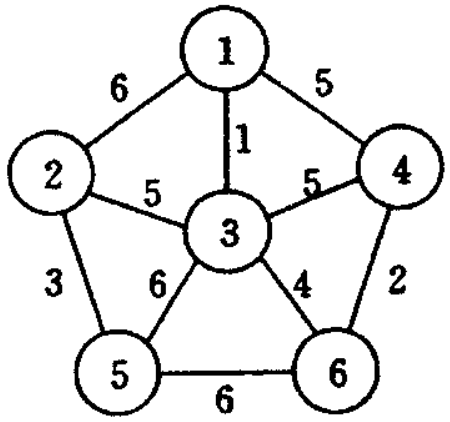
- 输入: G=(V,E,w)无向连通; 输出: G的MST
- 维护U⊆V, 初始任取V中一点r

当U!=V, 取U到V-U的最小权边(u,v), 将v加入U.

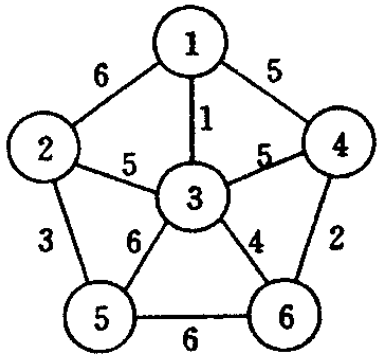
key[u]记u到U的最小距离, G[u]记u与U最小距离对应点

1. 初始G[u]=NIL, key[r]=0, 其它key[u]=INF, Q=V, U空
2. 当Q非空
3. 取出Q中u使得key[u]最小, 加入U
4. 对u的每个邻居v,
5. 若 v∈Q 且 w[u,v] < key[v]
6. 则 key[v] = w[u,v], G[v] = u

Q一般数组O(|V|²+|E|) = O(|V|²)



Prim算法O(|E|log|V|)



- 输入: G=(V,E,w)无向连通; 输出: G的MST
- 维护U⊆V, 初始任取V中一点r

当U!=V, 取U到V-U的最小权边(u,v), 将v加入U.

key[u]记u到U的最小距离, G[u]记u与U最小距离对应点

1. 初始G[u]=NIL, key[r]=0, 其它key[u]=INF, Q={r}, U空
2. 当优先队列(最小堆)Q非空
3. 删除Q堆顶元素u. 若u在U中,则break; 否则加入U.
4. 对u的每个邻居v,
5. 若 w[u,v] < key[v]
6. 则 将v按键值key[v]=w[u,v]插入Q, G[v] = u

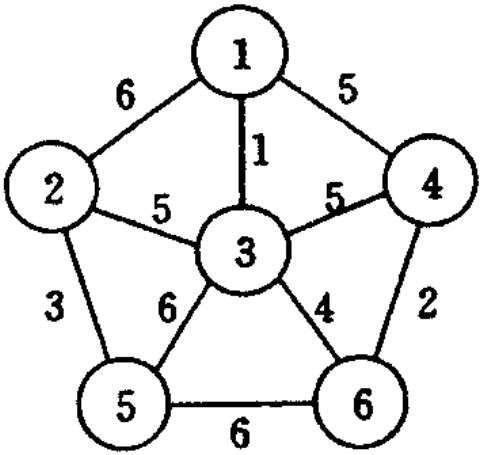
Q优先队列O(|V|log|E|+|E|log|E|) = O(|E|log|E|)

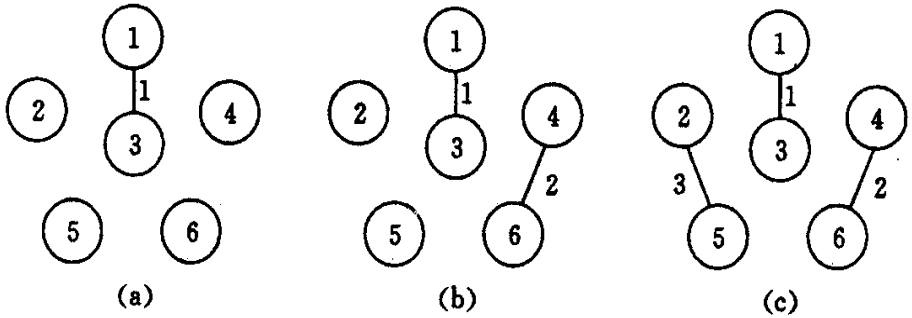
Kruskal算法

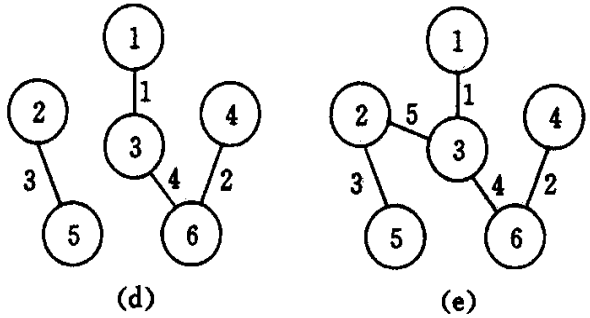
- 输入: G=(V,E,w)无向连通; 输出: G的MST
- 维护A,Q⊆E, 初始A为空集, Q=E升序

Q = { 13, 46, 25, 36, 34, 23, 14, 12, 35, 56 }

当Q非空, 取Q最小权边(u,v), 若可行, 加入A.



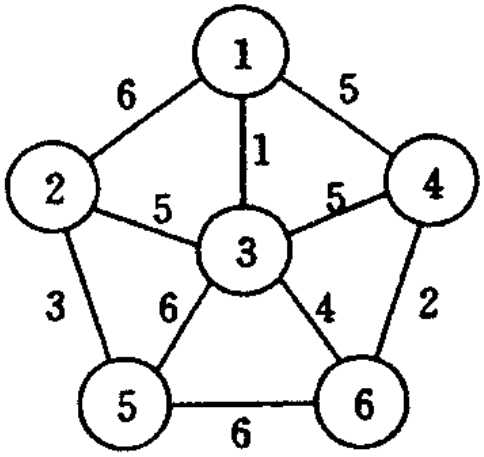




Kruskal算法细节O(|E|log|V|)

- 输入: G=(V,E,w)无向连通; 输出: G的MST
- 维护A,Q⊆E, 初始A为空集, Q=E升序

当Q非空, 取Q最小权边(u,v), 若可行, 加入A.



1. A为空, Q=E按边权升序排列
2. 当Q非空
3. 顺序取Q中边(u,v)
4. 若u,v在不同连通分支(检查),
5. 则添(u,v)到A, 合并u,v所在连通分支,
6. 输出A

并查集算法处理连通分支, O(|E|log|E|+|E|log*|V|)

并查集算法(Make-set, Find-set)

p[x]: x的父亲; rank[x]: x的阶; Find(x): 找x的根

```
Make(x)
1 p[x]=x
2 rank[x]=0
```

```
Union(x,y)
1 Link(Find(x), Find(y))
```

```
Find(x)
1 若 x≠p[x], 则
2   p[x] = Find(p[x])
3  返回 p[x]
```

路径压缩(pc)技术

```
Link(x,y) //合并两树根
1 若 rank[x]>rank[y]
2 则 p[y]=x
3 否则 p[x]=y
4     若 rank[x]=rank[y]
5     则 rank[y]=rank[y]+1
```

性质1:
只有树根
rank值
会增加

性质2:
树根阶
= r
⇓
树节点数
$\geq 2^r$.

n个节点, m次操作. 不计pc: O(mlogn); 计pc: O(m α(n))

回顾定义(b>1)2^^b=$2^{2^{\wedge\wedge(b-1)}}$, 2^^1=2, log*(2^^b) = b

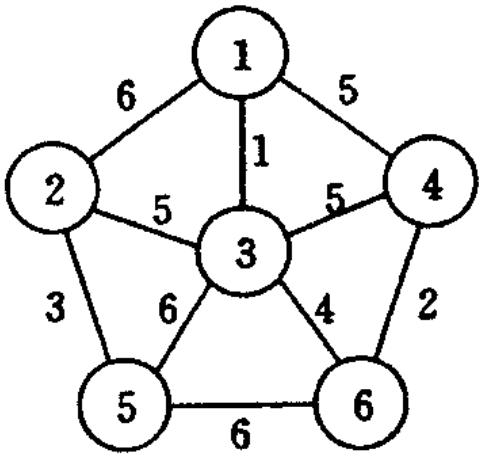
加入并查集结构的Kruskal算法

1. A为空, Q=E按边权升序排列, 每个点是一颗树
2. 当Q非空
3. 顺序取Q中边(u,v)
4. 若u,v在不同树中, 则添(u,v)到A, 合并u,v所在树,
5. 输出A

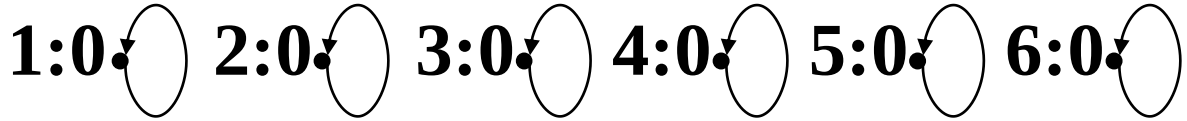
---

1. A为空, Q=E按边权升序排列, ∀x Make(x)
2. 当Q非空
3. 顺序取Q中边(u,v)
4. 若Find(u)≠Find(v), 则添(u,v)到A, Union(u,v),
5. 输出A

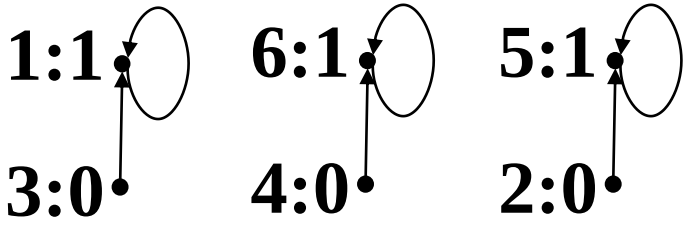
Kruskal: 取边, 查找, 合并



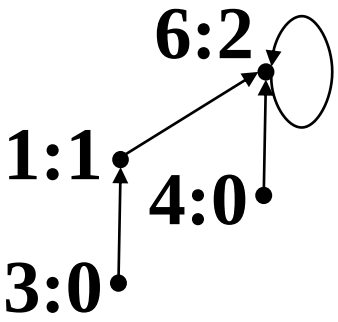
Q={13, 46, 25, 36, 34, 23, 14, 12, 35, 56}



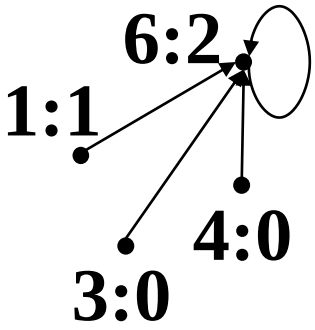
查找合并
(1,3) (4,6) (2,5)



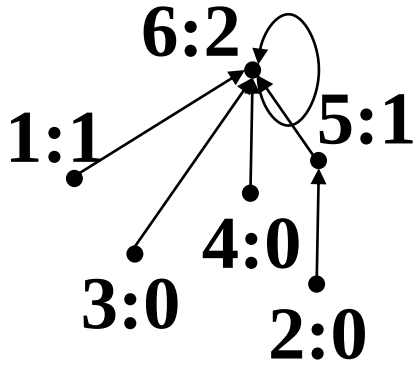
查并(3,6)



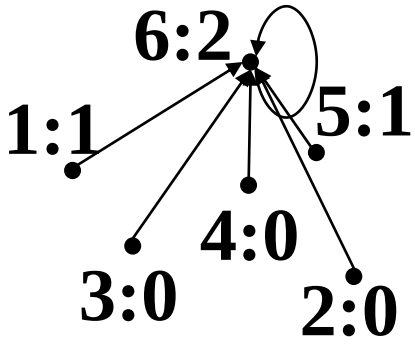
查(3,4)



查并(2,3)



查(1,2)



[刘]n个节点, m次操作, O((n+m)log*n)

n个节点分log*n块: 第b块={x: 2^^(b-1)<rank[x]≤2^^b }

性质1: 只有树根rank会增加, 非树根rank不变.

性质2: 树根阶 = r ⇒ 树节点数 ≥ 2^r.

性质3:至多有$n/2^r$个rank为r的点, rank值不大于n-1

性质4: 第b块至多n/(2^^b)个点.

设Find(x_0)依次经过$x_0, x_1, \ldots, x_{L-1}$到根节点$x_L$. x_i分四类

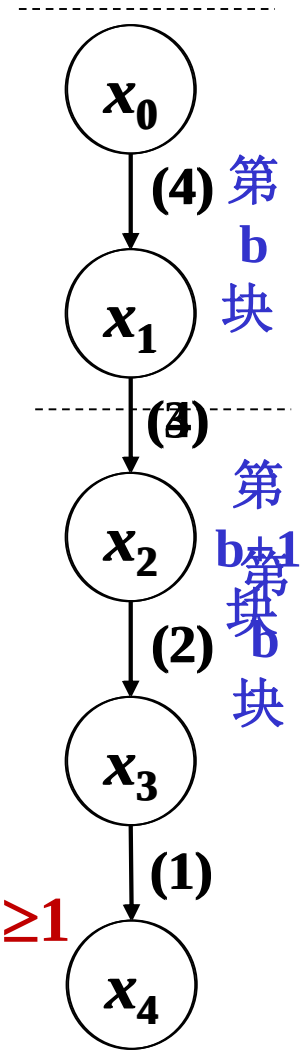
(1)根 (2)根的儿子 (3)非根或根的儿子, x_i和父不在同块

(4) 非根或根的儿子, x_i和父在同块

Find(x_0): (1)贡献1次; (2)贡献1次; (3)贡献≤ log*n次.

(4)贡献:求m次操作总和的上界 //x_i:阶不变,父变, 父阶增≥1

- 设x_i和父都在第b块, 则x_i对(4)的贡献 ≤ 2^^b 次;
- 第b块所有元素对(4)的贡献 ≤ (n/2^^b) × 2^^b = n 次.



第四章 贪心算法
Greedy

!!贪心不一定正确(0-1背包), 需要证明

1. 活动安排问题
2. 贪心算法基本要素
3. Huffman算法
4. 最小生成树
5. 多机调度问题

多机调度问题

输入: n,m, t[1:n], n个作业, m台机器

输出: 最早完成全部作业的调度方案.

要求: 作业不能拆分

Multiprocessor scheduling, NP完全, 现无有效的解法

近似算法: LPT(Longest Processing Time)算法

$n \le m$: 一个机器一个作业

$n > m$: 作业按时长降序排列, 依次分配空闲机器

近似比例 $\le 4/3-1/(3m)$. 例m=1, m=2(11,10,7,7,7)

PTAS: $\forall \varepsilon>0$, 有$(1+\varepsilon)$-优化的多项式时间近似算法

APX: $\exists c>1$, 有c-优化的近似算法. 若$P \neq NP$, $PTAS \subset APX$

Gone fishing

在一条水平路边, 有n个钓鱼湖, 从左到右编号1-n.

John有H小时的时间, 希望钓到尽量多的鱼.

从湖1出发向右, 在一些湖边停留一定时间钓鱼.

求钓最多鱼的方案(多余时间都给湖1).

(i=1:n)在第i个湖中钓鱼, 第1个5分钟可以钓到F[i]条鱼,

以后每钓5分钟鱼, 可钓到的鱼数减少D[i].

(i=1:n-1)从第i个湖到第i+1个湖需要5×T[i]分钟.

输入:

```
n    //n=0输入结束
H
F[1:n]
D[1:n]
T[1:n-1]
n …
```

```
2
1
10 1
2 5
2
```

```
45, 5
Number of fish expected: 31
```

1	2	3	4	5	6	7	8	9	10	11	12	
1	1	1	1	1	1	1	1	1	→	→	2	
10	8	6	4	2	0	0	0	0	-	-	1	合计31

本章小结

贪心不一定正确, 需要证明

1. 活动安排问题
2. 贪心算法基本要素
3. Huffman算法
4. 最小生成树
5. 多机调度问题

本章作业

1. 字符a~h出现的频率恰好是前8个Fibonacci数, 它们的Huffman编码是什么? 将结果推广到n个字符的频率恰好是前n个Fibonacci数的情形.
2. 若在0-1背包问题中, 各物品依重量递增排列时, 其价值恰好降序排列, 对这个特殊的0-1背包问题, 设计一个有效算法找出最优解, 并说明算法的正确性.
3. 将最优装载问题的贪心算法推广到2艘船的情形, 贪心算法还能产生最优解吗?
4. 最优分解问题. 见下页

本章作业

4. 最优分解问题.

问题描述:设n是一个正整数,将n分解为若干互不相同的自然数之和,且使这些自然数的乘积最大.

算法设计:对于给定的正整数n,计算最优分解方案.

数据输入:由文件input.txt提供输入数据.

文件只有一行,是正整数n.

结果输出:将计算的最大乘积输出到文件output.txt

例如若n=10, 则最优分解为2+3+5, 最大乘积为30.

附录一

最小堆自底向上建堆Θ(n)

数据结构

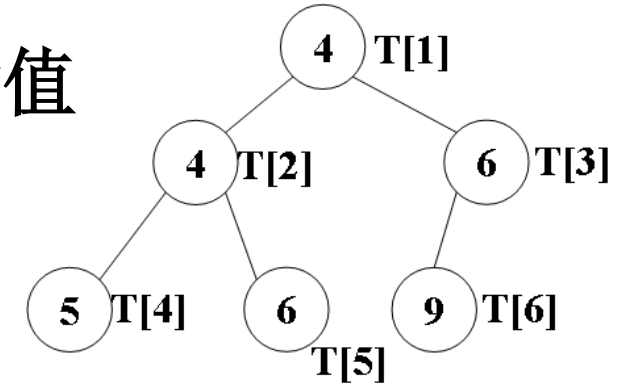
- 数组, 链表, 二叉树(堆和搜索树), hash表
- (最小)堆: 每个节点的关键值

  小于或等于左右子节点的关键值

  操作: 插入, 删除, O(h)

  一般隐式存储, 即数组

  T[i]的左孩子T[2i], 右孩子T[2i+1]



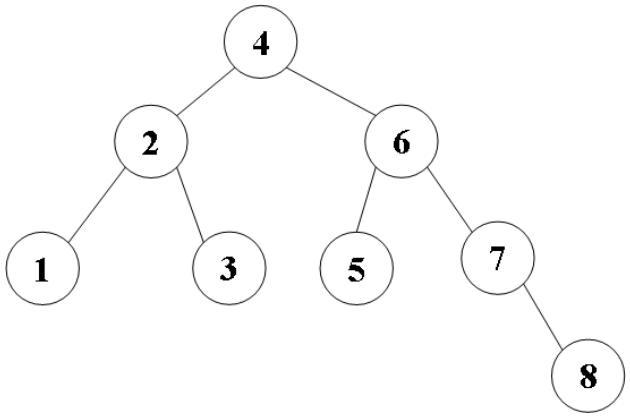
4	4	6	5	6	9

- 二叉搜索树: 每个节点的关键值

  大于所有左分支节点关键值,

  小于所有右分支节点关键值

  操作: 查找, 插入, 删除, O(h)

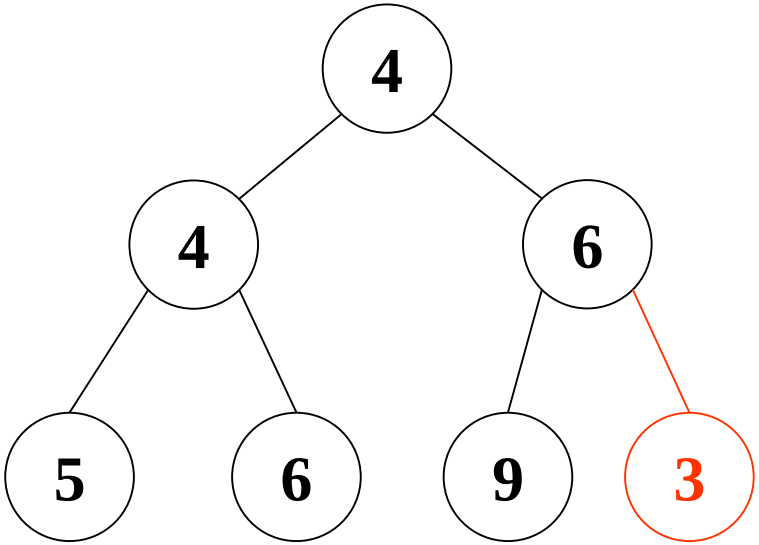


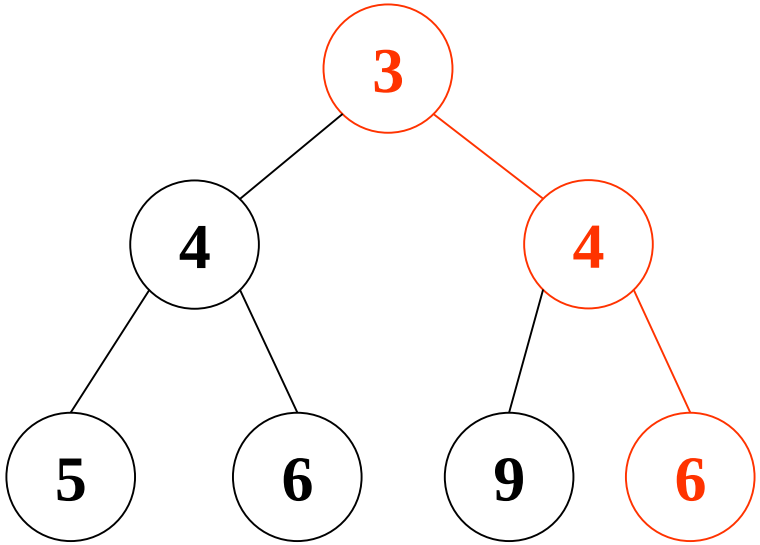
最小堆的插入操作

例如: 在下面的堆中插入3

过程: 先放在最后, 再调整

1	2	3	4	5	6	7
4	4	6	5	6	9	3
4	4	3	5	6	9	6
3	4	4	5	6	9	6





堆的删除操作

例如: 对下面的堆中作删除

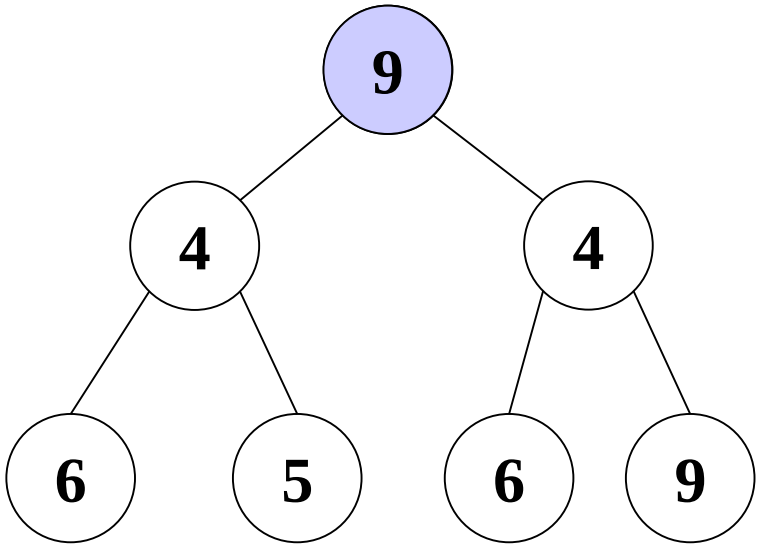
过程: 取出T[1],

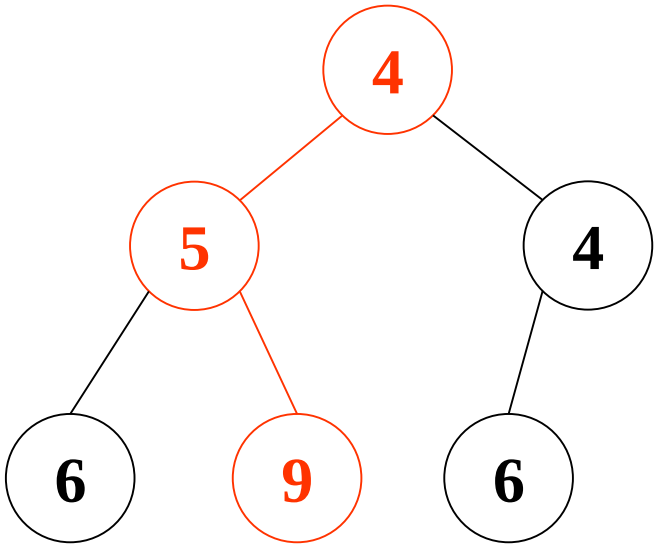
　　将T[n]赋给T[1], 调整

特点: 仍然是连续片段

- 堆顶以外元素位置很乱

1	2	3	4	5	6	7
3	4	4	6	5	6	9
9	4	4	6	5	6	
4	9	4	6	5	6	
4	5	4	6	9	6	





自顶向下建堆

	1	2	3	4	5	6	7	8
1	4	2	6	1	3	5	7	8
2	2	4	6	1	3	5	7	8
3	2	4	6	1	3	5	7	8
4	1	2	6	4	3	5	7	8
5	1	2	6	4	3	5	7	8
6	1	2	5	4	3	6	7	8
7	1	2	5	4	3	6	7	8
8	1	2	5	4	3	6	7	8

$$\sum_{i=1}^{n} \log i \geq \frac{n}{2} \log \frac{n}{2}$$

$$\sum_{i=1}^{n} \log i = \Theta(n \log n)$$

自底向上建堆

	1	2	3	4	5	6	7	8
1	4	2	6	1	3	5	7	8
2	4	2	6	1	3	5	7	8
3	4	2	6	1	3	5	7	8
4	4	2	6	1	3	5	7	8
5	4	2	6	1	3	5	7	8
6	4	2	5	1	3	6	7	8
7	4	1	5	2	3	6	7	8
8	1	2	5	4	3	6	7	8

$$T(n) = 2T(n/2) + \log n = \Theta(n)$$

本质上是分治(奇怪的分)

用数学归纳法容易证明:

$T(n) \le c_2 n - 2\log n$

$T(n) \ge c_1 n$

附录二

活动安排问题的动态规划算法

活动安排问题的动态规划算法

n个活动申请一个活动室, 活动起始终止区间(s_i, f_i)

- 输入: n, (s_i, f_i), $i = 1{:}n$, 输出: 最大相容活动子集
- 设 $f_1 \le f_2 \le \ldots \le f_n$, 添加 $f_0 = 0$, $s_{n+1} = \infty$
- S[i,j] = {在活动i结束后和活动j开始前能添加的活动}
- c[i,j] = S[i,j]中的最大相容活动子集
- 输出 = c[0, n+1]
- 若i ≥ j, 则S[i,j] = ∅.

$$c[i,j] = \begin{cases} 0 & \text{若} S[i,j] = \phi \\ \max\limits_{i<k<j}\{c[i,k] + c[k,j] + 1\} & \text{否则} \end{cases}$$