一、活动选择问题

> 一个大厅，同时只能一个活动，最多能举办多少
> 每个活动ai开始时间Si，结束Fi

贪心策略：

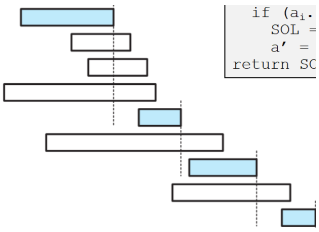
- 把所有活动按结束时间排序
- 第一个结束的活动一定要选

> 反证法易证

- 选择与之前不冲突，且第一个结束的活动

> S' are the activities starting after a'
> 则OPT(S)=OPT(S')∪{a'}.

```
ActivitySelection(S):
Sort S into increasing order of finish time
SOL = {a1}, a’ = a1
for (i=2 to n)
  if (ai.start_time > a’.finish_time)
    SOL = SOL ∪ {ai}
    a’ = ai
return SOL
```



二、贪心算法原理

最优子结构

> Optimal substructure

- 一个问题的最优解包含其子问题的最优解

贪心步骤

```
分解问题：一次选择+子问题
证明贪心选择安全，即原问题可以做出一次贪心选择
证明贪心选择与子问题组合可以得到最优解
```

> 在每个贪心算法之下，几乎总有一个更繁琐的动态规划算法，

Optimal substructure [usually easy to prove]: optimal solution to the problem contains within it optimal solution(s) to subproblem(s).

Greedy choice [could be hard to identify and prove]: the greedy choice is contained within some optimal solution.

三、贪心效果

```
arbitrarily bad solutions: 0-1 knapsack, …
optimal solutions: MST, Huffman codes
near-optimal solutions: Set cover, …
```

例一、小偷装包问题

> 贪心不解决问题， arbitrarily bad
>
> 每件东西有价值vi和重量wi，小偷能带走的总重量有限

物品可分拆

- 贪心：每次拿max(vi/wi)

- **Lemma 1 [greedy-choice]:** let a_m be a most cost efficient item, then in some optimal solution, at least $w'_m = \max\{w_m, W\}$ pounds of a_m are taken.
- **Proof:**
- Consider an optimal solution, assume $w' < w'_m$ pounds of a_m are taken.
- Now, substitute $w'_m - w'$ pounds of other items with a_m.
- Since a_m is the most cost-efficient, the new solution cannot be worse.
- **Lemma 2 [optimal substructure]:** let a_m be a most cost efficient item in A, then "$OPT_{W-\max\{w_m,W\}}(A - a_m)$ with $\max\{w_m, W\}$ pounds of a_m" is an optimal solution of the problem.
- **Proof:**
- Consider some $OPT_W(A)$ containing $\max\{w_m, W\}$ pounds of a_m.
- If optimal substructure does not hold, then $OPT_W(A)$ gives $SOL_{W-\max\{w_m,W\}}(A - a_m) > OPT_{W-\max\{w_m,W\}}(A - a_m)$.
- But this contradicts the optimality of $OPT_{W-\max\{w_m,W\}}(A - a_m)$.

0-1背包：物品不可分拆

- 贪心得到任意差的结果

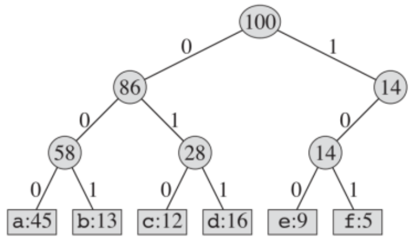
例二、optimal code tree

> 贪心给出最优解optimal solutions
>
> full二叉树表示，0或2孩子

- 总cost

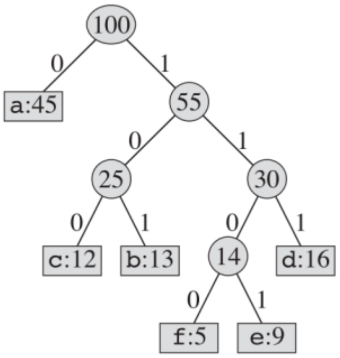
$$\sum_{i=1}^{n} f_i \cdot d_T(i) = \sum_{u \text{ in tree}\backslash\text{root}} f_u$$

- 等长编码



- prefix-free code

> **Huffman编码**



Huffman编码

> 对字符集大小归纳：
> 合并两个频率最小的结点

```
Huffman(C):
Build a priority queue Q based on frequency
for (i=1 to n-1)
  Allocate new node z
  x = z.left = Q.ExtractMin()
  y = z.right = Q.ExtractMin()
  z.frequency = x.frequency + y.frequency
  Q.Insert(z)
return Q.ExtractMin()
```

- O（nlgn）
- 最小的两个点xy一定是兄弟且深度最深

> 反设ab最深，深度问d，且不是xy，
> 交换a和x，得到T'

> 再交换b和y，得到T''，得到的cost更小

$$
\begin{aligned}
cost(T') &= cost(T) + \big(d - d_T(x)\big) \cdot f_x - \big(d - d_T(x)\big) \cdot f_a \\
&= cost(T) + \big(d - d_T(x)\big) \cdot (f_x - f_a) \le cost(T)
\end{aligned}
$$

- 将Tz中的z向下分成两个孩子xy得到的新树T是OCT

> C_z=C−{x,y}+{z} with f_z=f_x+f_y
>
> Tz表示Cz对应的树

Let T' be an optimal code tree for C, with x and y being sibling leaves.

$$
\begin{aligned}
cost(T') &= f_x + f_y + \textstyle\sum_{u \in T' \setminus \text{root and } u \notin \{x,y\}} f_u = f_x + f_y + cost(T'_z) \\
&\ge f_x + f_y + cost(T_z) = cost(T)
\end{aligned}
$$

So T must be an optimal code tree for C.

例三、最小覆盖

> 贪心得出接近的答案near-optimal solutions
>
> 以结点为圆心，r为半径画最少的圆，覆盖所有结点

- 策略：每次选择能覆盖最多点的圆
- 效果：如果最优需要k次，贪心策略上界klnn次

Proof: Let n_t be number of uncovered elements after t iterations. (Thus $n_0 = n$.)

These n_t elements can be covered by some k sets. (The optimal solution will do.)

So one of the remaining sets will cover at least n_t/k of these uncovered elements.

Thus $n_{t+1} \le n_t - n_t/k = n_t(1 - 1/k)$

$n_t \le n_0(1 - 1/k)^t < n_0\big(e^{-1/k}\big)^t = n \cdot e^{-t/k}$ $1 - x < e^{-x}$ when $x \ne 0$

With $t = k \ln n$ we have $n_t < 1$, by then we must have done!