中国科学技术大学 计算机科学与技术系
University of Science and Technology of China
DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

算法设计与分析
Design and Analysis of Algorithms

主讲人　徐云

Fall 2018, USTC

第16章 贪心算法

16.1 方法概述

- 基本思想
- 求解步骤
- 适合求解问题的特征
- 存在的问题
- 与动态规划法的比较
- 示例

基本思想

- 从问题的某一个初始解出发，通过一系列的贪心选择——当前状态下的局部最优选择，逐步逼近给定的目标，尽可能快地求得更好的解。
- 在贪心算法（greedy method）中也采用逐步构造最优解的方法。在每个阶段，都作出一个按某个评价函数最优的决策，该评价函数最优策略称为贪心准则（greedy criterion）。
- 贪心算法的正确性，就是要证明按贪心准则求得的解是全局最优解。
- 贪心算法不能对所有问题都得到全局最优解。但对许多问题它能产生全局最优解，如单源最短路经问题，最小生成树问题等。

求解步骤

从问题的某一初始解出发；

while 依据贪心策略朝给定目标前进一步 do

求出可行解的一个解元素；

由所有解元素组合成问题的一个可行解；

适合求解问题的特征

- 贪心选择性质：可通过局部最优（贪心）选择达到全局最优解；
  - 通常以自顶向下的方式进行，每次选择后将问题转化为规模更小的子问题；
  - 该性质是贪心法使用成功的保障，否则得到的是近优解；
- 最优子结构性质：问题的最优解包含它的子问题的最优解；
  - 并不是所有具有最优子结构性质的问题都可以采用贪心策略；
  - 往往可以利用最优子结构性质来证明贪心选择性质；

存在的问题

如果不满足贪心选择性质，贪心算法存在：

- 不能保证求得的最后解是最优的；
- 只能求满足某些约束条件范围的局部最优解。

注：贪心算法虽不能保证得到最优结果，但对于一些除了“穷举”方法外没有有效算法的问题，用贪心算法往往能很快地得出较好的结果，此方法还是很实用的。

与动态规划法的比较

Dynamic Programming	Greedy Algorithms
At each step, the choice is determined based on solutions of subproblems.	At each step, we quickly make a choice that currently looks best. -A local optimal (greedy) choice.
Sub-problems are solved first.	Greedy choice can be made first before solving further sub-problems.
Bottom-up approach	Top-down approach
Can be slower, more complex	Usually faster, simpler

示例（1）

- 例1：两种背包问题

  (1)0-1背包问题

  (2)小数背包问题

两种背包问题都满足最优子结构性质，都可以用动态规划来求解；

小数背包问题还具有贪心选择性质，用贪心法求解更简单、更快速（见16.2节）；但0-1背包问题用贪心法求解不一定能得到最优解；

示例（2）

- 例2：找钱问题：用最少的货币数找出钱A

(1)货币数量和种类不限制情形：具有贪心选择性质，可以使用贪心法：按货币单位从高往低给付，总能得到最优解（见16.5节）。

(2)货币数量和种类有限制情形：贪心法并不总能得到最优解

设n个货币$P=\{p_1,p_2,\ldots,p_n\}$，d_i和x_i分别是p_i的货币单位和选择的数量，问题的形式描述为：

$$\min\left\{\sum_{i=1}^{n} x_i\right\}$$

$$\text{s.t.}\quad \begin{cases} \sum_{i=1}^{n} d_i x_i = A \\ x_i = 0,1 \end{cases}$$

示例（3）

- 例2：找钱问题：（Cont.）

(2)货币数量和种类有限制情形

– 穷举法

解空间$X=\{(x_1,x_2,\ldots,x_n)|x_i\in\{0,1\},i=1,\ldots,n\}$ $\therefore |X|=2^n$

对任意$x\in X$，判断x是否是可行解（即满足约束条件）的时间为$O(n)$。

因此，穷举法的时间复杂度$T(n)=O(n2^n)$

– 贪心法

如问题：8个硬币(5个1分,2个1角,1个1角5分)，数出2角

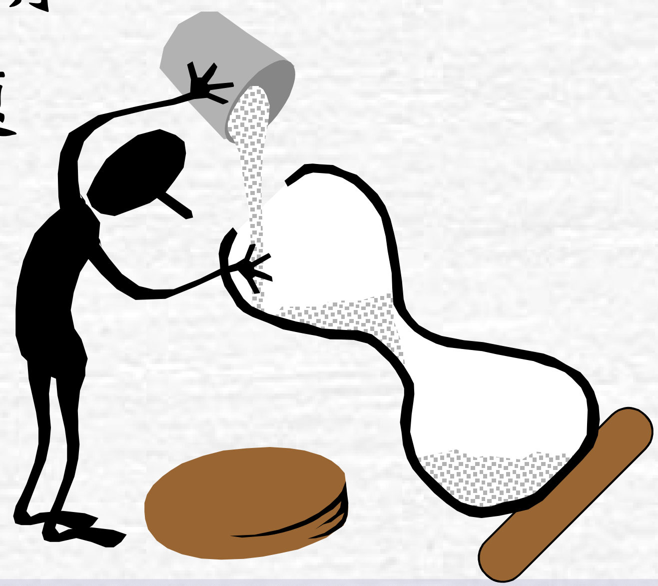
按货币单位降序数钱（贪心策略）：$1\times15+5\times1=2$角，

显然不是最优解。

第16章 贪心算法

16.2 小数背包问题

- 问题描述及示例
- 贪心策略设计
- 贪心算法
- 贪心选择性质的证明
- 特殊的0-1背包问题



问题描述及示例

- 问题描述

$$\max\sum_{i=1}^{n} v_i x_i$$

x_i为装入物品i的比例

$$\begin{cases}\sum_{i=1}^{n} w_i x_i \le c & w_i > 0 \\ 0 \le x_i \le 1 & i = 1,2,\ldots,n \\ v_i > 0,\ w_i > 0,\ c > 0 & i = 1,2,\ldots,n\end{cases}$$

w_i为重量，c为背包容量

v_i为价值

v_i / w_i为价值率(单位重量价值)

- 示例： n=3,c=20,v=(25,24,15),W=(18,15,10)，列举4个可行解

	(x_1, x_2, x_3)	$\sum w_i x_i$	$\sum v_i x_i$
①	(1/2, 1/3, 1/4)	16.5	24.5
②	(1, 2/15, 0)	20	28.2
③	(0, 2/3, 1)	20	31
④	(0, 1, 1/2)	20	31.5（最优解）

贪心策略设计

- 第一种策略：按价值最大贪心，使目标函数增长最快

  按价值排序从高到低选择物品=>②解(次最优)

- 第二种策略：按重量最小贪心，使背包增长最慢

  按重量排序从小到大选择物品=>③解(次最优)

- 第三种策略：按价值率最大贪心，使单位重量价值增长最快

  按价值率排序从大到小选择物品=>④解(最优)

贪心算法

```
GreedyKnapsack( n,  M, v[], w[], x[] )
{//按价值率最大贪心
    Sort(n, v, w);    //使v1/w1≥v2/w2≥…≥vn/wn
    for i=1 to n do x[i]=0;
    c=M;
    for i=1 to n do
    {   if(w[i]>c)  break;
        x[i]=1;
        c-=w[i];
     }
     if(i<=n)  x[i]=c/w[i];      //物品i是选择的最后一项
 }
```

$T(n)=O(n\log n)$

贪心选择的最优性证明（1）

- 定理：如果$v_1/w_1 \geq v_2/w_2 \geq \ldots \geq v_n/w_n$，则GreedyKnapsack算法对于给定的背包问题实例生成一个最优解
  - 证明的基本思想

    把贪心解与任一最优解相比较，如果这两个解不同，就去找开始不同的第一个X_i，然后设法用贪心解的X_i去代换最优解的X_i，并证明最优解在分量代换之后其总价值保持不变，反复进行下去，直到新产生的最优解与贪心解完全一样，从而证明了贪心解是最优解。

贪心选择的最优性证明（2）

– 证明：设$(x_1, \ldots ,x_n)$是贪心算法求得的解

$Case1$: 所有$x_i = 1$。显然该解就是最优解。

$Case2$: 设$X = (1,\ldots,1, x_j, 0,\ldots,0)$ $x_j \neq 1, 1 \le j \le n$。下证$X$就是最优解。设问题的最优解$Y = (y_1,\ldots,y_n)$，则存在$k$使得$y_k \neq x_k$的最小下标（否则$Y = X$，得证）。

首先，可以证明$y_k < x_k$。（反证：若$y_k > x_k$，则$x_k \neq 1, \therefore k \ge j$

$$\Rightarrow \sum_{i=1}^{n} w_i y_i \ge \sum_{i=1}^{k} w_i y_i > \sum_{i=1}^{k} w_i x_i \ge \sum_{i=1}^{j} w_i x_i = c \therefore Y\text{不是可行解，矛盾）}$$

下面改造Y成为新解$Z = (z_1,\ldots,z_n)$，并使Z仍为最优解。将y_k增加到x_k，从$(y_{k+1},\ldots,y_n)$中减同样的重量使总量仍是c。即，

$$z_i = x_i \quad i = 1,2,\ldots,k; \quad \text{和} \quad w_k(z_k - y_k) = \sum_{i=k+1}^{n} w_i(y_i - z_i)$$

贪心选择的最优性证明（3）

$$\therefore \quad \sum_{i=1}^{n} w_i z_i = \sum_{i=1}^{k-1} w_i z_i + w_k z_k + \sum_{i=k+1}^{n} w_i z_i$$

$$= \sum_{i=1}^{k-1} w_i y_i + w_k z_k + \left(\sum_{i=k+1}^{n} w_i y_i - w_k (z_k - y_k) \right) = \sum_{i=1}^{n} w_i y_i = c$$

$$\therefore \quad \sum_{i=1}^{n} v_i z_i = \sum_{i=1}^{n} v_i y_i + (z_k - y_k) v_k - \sum_{i=k+1}^{n} (y_i - z_i) v_i \quad // \because Z\text{由}Y\text{变得的}$$

$$= \sum_{i=1}^{n} v_i y_i + (z_k - y_k) w_k v_k / w_k - \sum_{i=k+1}^{n} (y_i - z_i) w_i v_i / w_i$$

$$\geq \sum_{i=1}^{n} v_i y_i + \left[(z_k - y_k) w_k - \sum_{i=k+1}^{n} (y_i - z_i) w_i \right] v_k / w_k \quad // \text{利用} v_i / w_i \downarrow$$

$$= \sum_{i=1}^{n} v_i y_i$$

$\therefore Z$也是最优解，且$z_i = x_i \; i = 1,\ldots,k$； 重复上面过程 $\Rightarrow$ X为最优解。

特殊的0-1背包问题

如果$w_1 \le w_2 \le \ldots \le w_n$， $v_1 \ge v_2 \ge \ldots \ge v_n$

$\Rightarrow v_1/w_1 \ge v_2/w_2 \ge \ldots \ge v_n/w_n$ 可以用贪心法求最优解

– 算法

```
0-1-Knapsack( v[], w[], n, c)
{ //输出x[1..n]
    for  i=1 to n do   x[i]=0;
    value = 0.0;
    for i=1 to n do
    {   if(w[i]<c)
          {  x[i]=1;    c-=w[i];  value+=v[i];  }
        else    break;
    }
    return value;
}
```

算法的正确性未证，

如何证明？

第16章 贪心算法

16.3 活动安排

- 问题描述
- 贪心策略及其最优化证明
- 算法描述
- 计算示例

问题描述

- 有n个活动集E={1,2,…,n}使用同一资源，而同一时间内同一资源只能由一个活动使用。每个活动的使用时间为$[s_i, f_i)$ i=1,…,n，s_i为开始时间，f_i为结束时间，若$[s_i, f_i)$与$[s_j, f_j)$不相交称活动i和活动j是相容的。
- 问题：选出最大的相容活动子集合。

注：该问题也可以用动态规划法求解但效率比贪心法低。

贪心策略及最优化证明

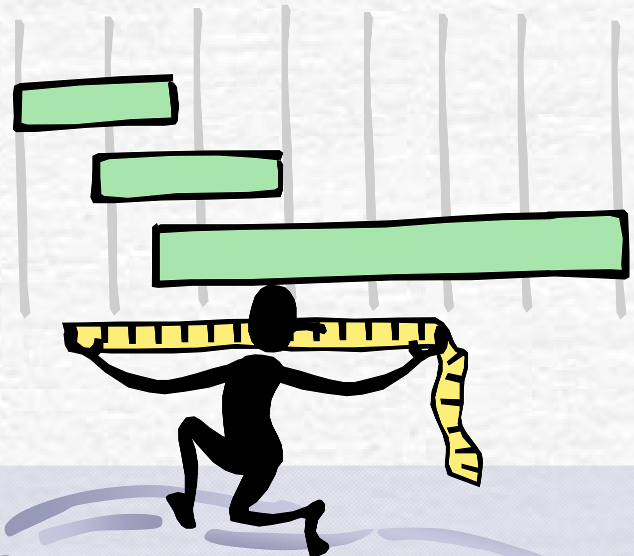
- 贪心策略

  设各活动已按结束时间排序：$f_1 \leq f_2 \leq \cdots \leq f_n$，先选出活动1，然后按活动编号从小到大的次序依次选择与当前活动相容的活动。

  注：这种策略使剩余的可安排时间极大化，以便于安排尽可能多的相容活动。

- 最优化证明（略）

  也就是贪心算法正确性证明



算法描述

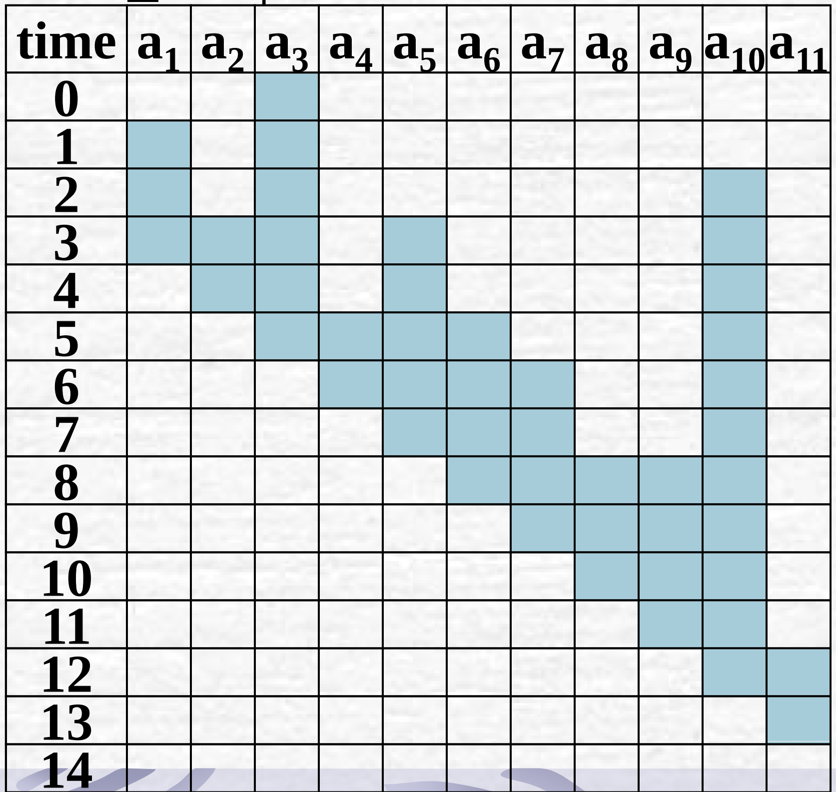
```
ActivitySelection( n, s[], f[], a[])
{//f[]已排序，a[]记录选择的活动，即a[i]=true表示活动i已选择
    a[1]=true;
    j=1;
    for i=2 to n do
    {   if(s[i]>=f[j])
          {  a[i]=true;
             j=i;}
        else  a[i]=false;
     }
 }
```

T(n)=O(nlogn)

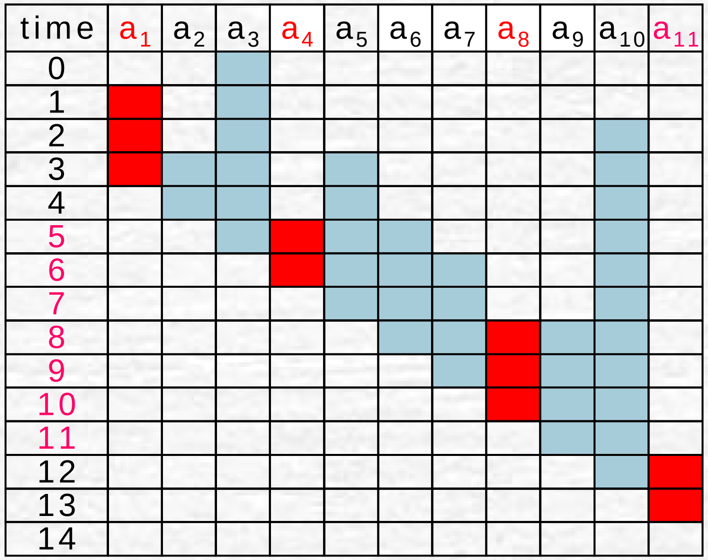
计算示例

11个活动已按结束时间排序，用贪心算法求解：

i	1	2	3	4	5	6	7	8	9	10	11
$start\_time_i$	1	3	0	5	3	5	6	8	8	2	12
$finish\_time_i$	4	5	6	7	8	9	10	11	12	13	14



相容活动：$\{a_3, a_9, a_{11}\}$, $\{a_1,a_4,a_8,a_{11}\}$, $\{a_2,a_4,a_9,a_{11}\}$



第16章 贪心算法

16.4 最优装载

- 问题定义
- 贪心策略
- 计算示例
- 算法描述
- 贪心算法最优性证明

问题定义

轮船载重为c，集装箱重量为$w_i\ (i=1,2,\ldots,n)$，在装载体积不受限制的情况下，尽可能多的集装箱装上轮船。

问题的形式化定义：

$$\max\sum_{i=1}^{n} x_i$$

$$s.t.\begin{cases}\sum_{i=1}^{n} w_i x_i \le c & w_i > 0\\ x_i \in \{0,1\} & i=1,2,\ldots,n\end{cases}$$

贪心策略

从剩下的货箱中，选择重量最小的货箱。这种选择次序可以保证所选的货箱总重量最小，从而可以装载更多的货箱。根据这种贪婪策略，首先选择最轻的货箱，然后选次轻的货箱，如此下去直到所有货箱均装上船或船上不能再容纳其他任何一个货箱。

计算示例

假设n=8, [w_1, ... ,w_8]=[100,200,50,90,150,50,20,80], c=400。

利用贪心算法时，所考察货箱的顺序为7，3，6，8，4，1，5，2。

货箱7，3，6，8，4，1的总重量为390个单位且已被装载，剩下的装载能力为10个单位，小于剩下的任何一个货箱。

在这种贪心解决算法中得到[x_1, ..., x_8] = [1，0，1，1，0，1，1，1]且$\sum x_i = 6$。

算法描述

```
ContainerLoading( x[], w[], c, n)
{ //x[i] =1当且仅当货箱i被装载，对重量按间接寻址方式排序
   new t[n+1];    //产生数组t，用于间接寻址
   IndirectSort(w, t, n);    //此时，w[t[i]]≤w[t[i+1]], 1≤i<n
   for i = 1 to n do  //初始化x
        x[i] = 0;
   for(i = 1; i <= n && w[t[i]] <= c; i++) {  //按重量次序选择物品
        x[t[i]] = 1;
        c = c- w[t[i]];
    } // c为剩余容量
   delete t[];   //删除数组t
}
```

$T(n)=O(n\log n)$

贪心算法最优性证明

- 证明思路

证明可以采用如下方式来证明贪心算法的正确性：

不失一般性，可以假设货箱都排好序：即$w_i \le w_{i+1}$ $(1 \le i \le n)$。

令$x=[x_1, \ldots, x_n]$为用贪心算法获得的解，

$y=[y_1, \ldots, y_n]$为一个最优解，

分若干步可以将y转化为x，转换过程中每一步都产生一个可行的新y，且$\sum_{i=1}^{n} y_i$值不变（即仍为最优解），便证明了X为最优解。

第16章 贪心算法

16.5 找钱问题

- 问题定义
- 贪心策略和算法
- 最优子结构性质
- 贪心选择性质

问题定义

使用2角5分，1角，5分和1分四种面值的硬币时（各种硬币数量不限），设计一个找A分钱的贪心算法，并证明算法能产生一个最优解。

设货币种类$P=\{p_1, p_2, p_3, p_4\}$，d_i和x_i分别是p_i的货币单位和选择数量，问题的形式描述为：

$$\min\left\{\sum_{i=1}^{4} x_i\right\}$$

$$s.t.\quad \begin{cases} \sum_{i=1}^{4} d_i x_i = A \\ x_i\text{为非负整数} \quad 1 \le i \le 4 \end{cases}$$

贪心策略和算法

- 贪心策略

$$x_1 = \lfloor A/d_1 \rfloor \qquad x_3 = \lfloor (A - d_1x_1 - d_2x_2)/d_3 \rfloor$$

$$x_2 = \lfloor (A - d_1x_1)/d_2 \rfloor \qquad x_4 = A - d_1x_1 - d_2x_2 - d_3x_3$$

- 算法

```
GreedyChange( d[], x[], A)
{//输出x[1..4]
    d[1]=25, d[2]=10, d[3]=5, d[4]=1;
    for i=1 to 3 do
    {   x[i]=A/d[i]; A-=x[i]*d[i];
     }
    x[4]=A;
 }
```

最优子结构性质

证明：

设$X=(x_1,x_2,x_3,x_4)$是问题钱数为A的最优解，则$X'=(0,x_2,x_3,x_4)$是子问题钱数为$A-d_1x_1$的最优解。

下面反证：若不然，$Y=(0,y_2,y_3,y_4)$是子问题钱数为$A-d_1x_1$的最优解，即Y优于X'

$$\sum_{i=2}^{4} y_i < \sum_{i=2}^{4} x_i \quad 且 \quad \sum_{i=2}^{4} d_i y_i = A-d_1x_1$$

$$\Rightarrow x_1+\sum_{i=2}^{4} y_i < \sum_{i=1}^{4} x_i \ 且\ d_1x_1+\sum_{i=2}^{4} d_i y_i = A$$

$$\Rightarrow (x_1,y_2,y_3,y_4)比(x_1,x_2,x_3,x_4)更优，矛盾。$$

贪心选择性质

证明：

设$X=(x_1,x_2,x_3,x_4)$是贪心解，$Y=(y_1,y_2,y_3,y_4)$是最优解.

可以证明：$x_1=y_1$

如果$x_1<y_1$，由$x_1=\lfloor A/d_1 \rfloor \Rightarrow x_1 \le A/d_1 < x_1+1 \Rightarrow d_1x_1 \le A <$

$d_1(x_1+1)$；因此，$\sum_{i=1}^{4} d_iy_i \ge d_1y_1 \ge d_1(x_1+1) > A$，产生矛盾；

如果$x_1>y_1$，则$10y_2+5y_3+1y_4 \ge 25$（否则，$25y_1+10y_2+5y_3+1y_4$

$<25(y_1+1) \le 25x_1+10x_2+5x_3+1x_4=A$，矛盾），因此用一个25分

的硬币替代等值的低于25分的硬币若干个（至少≥ 3），于是y不

是最优解；

综上，只能$x_1=y_1$；类似可以继续证得$x_2=y_2$，$x_3=y_3$，$x_4=y_4$；

$\therefore X$是最优解

End of Chap16