贪心算法

原理

贪心算法总是在当前步骤上选取最好的方案，即它是一种局部最优的选择，并希望它导致一个全局最优，但有时是不可能导致全局最优。

但是，仍有许多问题贪心法将产生全局最优解，如MST，单源最短路径等。

实例

活动选择问题

n设有n个活动$S = a_1, a_2 \ldots, a_n$，均要使用某资源(如教室)，该资源使用方式为独占式，一次只供一个活动使用。

- v每个活动a_i发生的时间为$[s_i, f_i), 0 \leq s_i < f_i < \infty$
- v两活动a_i, a_j相容（不冲突）是指：$[s_i, f_i), [s_j, f_j)$不重叠，满足$s_i \geq f_j$ or $s_j \geq f_i$。 即：一活动的开始时间大于等于另一活动的完成时间。

v活动选择问题：选择最多的互不冲突的活动，使相容活动集合最大。即：$A \subseteq S$，A中活动互不冲突且$|A|$最大。

动态规划

子问题设置

设子问题$S_{ij} \neq \emptyset, a_k \in S_{ij}, f_i \leq s_k \leq f_k \leq s_j$。若$S_{ij}$的解中选择了$a_k$，使用$a_k$产生$S_{ij}$的两个子问题

- S_{ik}包含在a_i完成之后开始，在a_k开始之前完成的那些活动
- S_{kj}包含在a_k完成之后开始，在a_j开始之前完成的那些活动

S_{ij}的解显然是S_{ik}和S_{kj}解的并，再加上a_k

最优子结构

设S_{ij}的最优解为A_{ij}，$a_k \in A_{ij}$，则用于计算的两个子问题的解也必须是最优的

递归形式

设$C[i, j]$表示S_{ij}的最优解的值，即S_{ij}中相容活动最大子集的size：$C[i, j] = |Aij|$

递推表达式如下

$$C[i,j]=\begin{cases}0 & S_{ij}=\emptyset \\ \max_{i<k<j}\{C[i,k]+C[k,j]+1\} & S_{ij}\neq\emptyset\end{cases}$$

贪心算法

设S_{ij}是任一非空子问题，a_m是S_{ij}中具有最早完成时间的活动，则

- 活动a_m必然被包含在S_{ij}中的某个最优解中
  - 子问题S_{im}是空集，使S_{mj}是唯一可能的非空集 (选a_m的目的)

因此我们可以每次取结束时间最早且不与前面所选活动冲突的活动即可获得最优解。

要点

- 将优化问题分解为做出一种选择及留下一个待解的子问题
- 证明对于原问题总是存在一个最优解会做出贪心选择，从而保证贪心选择是安全的
- 验证当做出贪心选择之后，它和剩余的一个子问题的最优解组合在一起，构成了原问题的最优解