高级算法设计与分析作业 5
贪心算法

2024.5.2

1

设纵向切割的位置从左到右分别为 a_1,a_2,a_3,a_4,a_5...a_m，横向切割的位置从上到下分别为 b_1,b_2,b_3...b_n。设 $A = \{a_i|i = 1, 2..., m\}$，$B = \{b_i|i = 1, 2..., m\}$。$C(x)$ 表示为选择 x 作为切割 x 序列的代价。则切割方案可以表示为包含 a_1,b_i 的序列，排在左侧的优先执行。

1.1

对于图中实例而言，我们有最小代价切割方案为：b_1,a_5,a_3,a_1,b_3,a_2,a_4,b_2。

总代价为 42。

1.2 自然语言描述

对横向和纵向的所有位置的执行代价从大到小进行排序，之后每次优先选择代价最高的位置进行切割。

1.3 方法正确性说明

贪心选择性设从大到小对横纵切割位置的执行代价进行排序，最优解每次优先选择执行代价最大的位置进行切割。

证明:

假设原命题不成立，则最优解可以表示为 $P^* = x'P_1x_0P_2$，其中 P_1、P_2 分别为由 $x \in A \cup B$ 所构成的序列，且 x' 位置的代价大于 x_0。故 $C(P^*) = C(\{x'\}) + C(P_1) + C(\{x_0\}) + C(P_2)$

令 $P' = x_0P_1x'P_2$。故该方案的代价为 $C(P') = C'(\{x_0\}) + C(P_1) + C'(\{x'\}) + C(P_2)$。

显然有 $C(x') + C(x_0) > C'(x_0) + C'(x')$，故 $C(P') < C(P^*)$。与 P^* 是最优解矛盾，故原命题成立。

优化子结构设优化解为 P^*，x_0 为切割代价最大的位置，则 $P^* - \{x_0\}$ 是 $A \cup B - \{x_0\}$ 问题的最优解。

证明：设问题的优化解为 P^*. 若原命题不成立，则 $P^* - \{x_0\}$ 不是子问题的优化解，存在 P'' 为子问题 $A \cup B - \{x_0\}$ 的优化解。

则总代价为 $C(P'') + C(\{x_0\}) < C(P^* - \{x_0\}) + C(\{x_0\}) = C(P^*)$ 与 P^* 是问题的最优解矛盾，故原命题成立。

Algorithm 1: 切割方案

```
Data: A, B
P = ∅;
W=sort_smallest_first(A ∪ B);
for i = 1 to W.length do
    P ← P ∪ W[i];
end
return P;
```

在考虑排序的情况下算法的时间复杂度为 $O((m+n)log(m+n))$，不考虑排序则为 $O(m+n)$

2

为了使得用户的平均等待时间 $1/n \sum e_i$ 达到最小，每次选择所需处理时间最短的任务进行处理，一个任务处理完后立刻进行下一个任务的处理，直到全部任务处理完成。

贪心选择性设 $A = 1, 2, ..., n$ 是 n 个任务的集合，假设所有任务都已经按照所需时间的从小到大进行排序，即 $a_1 \le a_2 \le ... \le a_n$，则存在 A 的调度问题的一个最优解即平均等待时间最短的调度，将时间最短的任务 1 安排在第一个处理。

证明：设 $1, 2, ..., n$ 的一个排列 $i_1, i_2...i_n$ 是问题的一个最优解，e_j 为第 j 个被处理的任务的等待时间。

若 $i_1 = 1$，贪心选择性成立。

若 $i_1 \ne 1$，假设任务 r 是第一个处理的任务 $(r \ne 1)$，而任务 1 排第 k 个处理，即调度为 $ri_2...i_{k-1}1i_{k+1}...i_n$，那么将任务 1 与任务 r 调换次序，得到新调度 $li_2...i_{k-1}ri_{k+1}...i_n$，其中 $e\ _j$ 为新的调度中第 j 个被处理的任务的等待时间。

显然有 $e_j = e\ _j, j = k+1, ..., n$。

由于 $a_1 \le a_r$，则有

$e\ _1 = a_1 \le a_r = e_1$,

$e\ _2 = a_1 + a_{i_2} \le a_r + a_{i_2} = e_2$,

...

$e\ _{k-1} = a_1 + a_{i_2} + ... + a_{k-1} \le a_r + a_{i_2} + ... + a_{k-1} = e_{k-1}$,

$e\ _k = a_1 + a_{i_2} + ... + a_{k-1} + a_r \leq a_r + a_{i_2} + ... + a_{k-1} + a_1 = e_k$,

因而，有 $1/n\sum_{j=1}^{n} e'_j \leq 1/n\sum_{j=1}^{n} e_j$，由于调度 $ri_2...i_{k-1}1i_{k+1}...i_n$ 是一个最优解，进而有 $1/n\sum_{j=1}^{n} e'_j \geq 1/n\sum_{j=1}^{n} e_j$，从而可以得到 $1/n\sum_{j=1}^{n} e'_j = 1/n\sum_{j=1}^{n} e_j$。所以调度 $li_2...i_{k-1}ri_{k+1}...i_n$ 也是问题的一个最优解，并且首先处理任务 1。

优化子结构设 $A = 1, 2, ..., n$ 是 n 个任务的集合，假设所有任务都已经按照所需时间的从小到大进行排序，即 $a_1 \leq a_2 \leq ... \leq a_n$。设 $li_2...i_{k-1}ri_{k+1}...i_n$ 是问题的一个最优解，则 $A' = A - \{1\}$ 一定是 $S' = S - \{1\}$ 的优化解。

证明：调度 S 中的每一项任务都包含在 A 中，所以 S 是 A 的一个解。

假设调度 $S = i_2...i_n$ 不是 A 的调度问题的最优解, 则存在另一个调度 $i'_2...i'_n$ 为问题的最优解，$1/n - 1\sum_{j=2}^{n} e_{i'_n} < 1/n - 1\sum_{j=2}^{n} e_{i_n}$.

那么 $1i\ _2...i\ _n$ 也是 A 的调度问题的一个解，且 $\frac{1}{n}e_1 + \sum_{j=2}^{n} e_{i'_n} = \frac{1}{n}a_1 + \sum_{j=2}^{n} e_{i'_n} < \frac{1}{n}a_1 + \sum_{j=2}^{n} e_{i_n} = \frac{1}{n}e_1 + \sum_{j=2}^{n} e_{i_n}$。与 $li_2...i_{k-1}ri_{k+1}...i_n$ 是问题的最优解矛盾，故结论成立。

Algorithm 2: Task-Plan

Data: $A, a_1, ..., a_n$

$n = A.length$;

for $i \leftarrow 1$ *to* n **do**

  $B[i] \leftarrow i$;

end

return B;

考虑排序算法的时间复杂度为 $O(nlogn)$，否则则为 $O(n)$

3

贪心思想：让油厂 1 把全部油给油站 1, 如有剩，则全给油站 2, 如还有剩, 则全给油站 3......

再让油厂 2 把全部油都给油站 1, 如有剩, 则全给油站 2, 如还有剩, 则全给油站 3......

贪心选择性若当前最左侧缺油的油站为 1，则问题的最优解需要依次遍历 1 以后的油站优先向 1 输油直到 1 满足。

证明: 假设原命题不成立, 存在一个更优的解 A, 且 A 是最优解, 那么 A 一定满足: 存在 a, b, c, d, x 使得油厂 a 的 x 量油给油站 d, 油厂 b 的 x 量油给油站 c, 且 $a < b$, $c < d$。设 $l(p, q)$ 表示位置 p 和 q 之间的距离。那么, 这 2x 量油的运输成本为 $x(l(a, d)+l(b, c))$, 而如果让 a 的 x 量油给 c, b 的给 d, 则成本为 $x(l(a, c)+l(b, d))$。显然 $x(l(a, d)+l(b, c)) > x(l(a, c)+l(b, d))$, 与 A 是最优解的假设矛盾。故原命题成立。

优化子结构假设当前最左侧的位置为 p，位置的集合为 P。问题的最优解为 A^*，则 $A^* - \{p\}$ 为问题 $P - \{p\}$ 的优化解。

证明：假设原命题不成立，则存在 $P-\{p\}$ 的优化解为 P'，使得 $Cost(P') < Cost(A^* - \{p\})$ 最优解的代价为 $Cost(A^*) = Cost(\{p\}) + Cost(A^* - \{p\}) < Cost(P') + Cost(\{p\})$ 与 P' 是最优解矛盾。故原命题成立。

Algorithm 3: 输油策略

Data: X, Y
$n \leftarrow X.length$;
$P = \emptyset$;
for $i \leftarrow 1$ *to* n **do**
    for $j \leftarrow 1$ *to* n **do**
        if $X[i] = 0$ **then**
            break;
        if $Y[j] > 0$ **then**
            if $Y[j] > X[i]$ **then**
                $P \leftarrow P \cup (i, j, X[i])$;
                $Y[j] \leftarrow Y[j] - X[i]$;
                $X[i] \leftarrow 0$;
            else
                $P \leftarrow P \cup (i, j, X[i] - Y[j])$;
                $Y[j] \leftarrow 0$;
                $X[i] \leftarrow X[i] - Y[j]$;
            end
        else
            continue;
        end
    end
end
return P;

时间复杂度为 $O(n^2)$

4

贪心思想：选择当前满足条件的订单中 a_i 最小的订单 (a_i,b_i)。

用集合 R= {C|C 为订单的集合且 C 中订单都能满足}，我们的目标就是找到 R 中长度最长的订单集合。

引理：假设 A 为最优解，A 中满足条件的订单调度中，第 k 个完成的订单为 (a_m,b_m)，

第 k+1 个完成的订单为 (a_l,b_l)，如果 b_m b_l，则调换订单 m 和订单 l 的顺序也能满足条件。

证明：用 t_i 表示在 A 中，第 i 个调度的任务。

由题意可知，$b_m \geq \sum_{i=1}^{k-1} a_{t_i} + a_m$ 和 $b_l \geq \sum_{i=1}^{k-1} a_{t_i} + a_m + a_l$。

将两个订单调换之后，我们可以得到 $b_m \geq b_l \geq \sum_{i=1}^{k-1} a_{t_i} + a_m + a_l$ 和 $b_l \geq \sum_{i=1}^{k-1} a_{t_i} + a_m + a_l > \sum_{i=1}^{k-1} a_{t_i} + a_l$。而在 A 中其余的订单都没有变化，所以命题得证。

贪心选择性

将收到的 n 个订单按照 a_i 排序，设 $a_1 \leq a_2 \leq ... \leq a_n$ 选择问题的某个最优解包括订单 (a_1,b_1)。

证明：假设 A 是一个最优解，如果 $(a_1, b_1) \in A$，则命题得证。

否则的话，设满足的第一个订单为 (a_k,b_k)，必然有 $a_1 \leq a_k$，A 中排在 k 订单之后的任意的订单 (a_j,b_j)，满足 $b_j \geq a_k + \sum_{i\ j} ai$。用 (a_1,b_1) 代替 (a_k,b_k) 的位置，我们可以得到一个新的解 $A' = (A - (a_k, b_k)) \cup (a_1, b_1)$，其中对于 A' 中排在 1 订单之后的任意订单 (a_j, b_j)，满足 $b_j \geq a_k + \sum_{i\ j} ai \geq a_1 + \sum_{i\ j} ai$，所以 A 也是满足条件的解，并且 |A|==|A |，所以 A 也是最优解。命题得证。

优化子结构 $A - (a_1, b_1)$ 是 $S = \{C \subseteq A - (a_1, b_1) : C \cup (a_1, b_1) \in R\}$ 的最优解。其中由引理可得，已知一个集合 S 能被满足，S 中的排序可以按照 b_i 的大小排序满足的。

证明：可以很容易得到 $\forall(a_i, b_i) \in (A - (a_1, b_1)) \Rightarrow (a_i, b_i) \in S$，所以 $(A - (a_1, b_1))$ 是 S 的一个解。

如果不是最优解，那么存在一个最优解 $B \ \in S$，使得 $|A - (a_1, b_1)| < |B'|$，那么我们可以得到一个集合 $A' = B' \cup (a_1, b_1)$，由已知条件很容易得到 A 为原问题的解，并且 $|A'| = |B\ | + 1 > |A - (a_1, b_1)| + 1 = |A|$，矛盾。

假设订单按照 A 排好序，则伪代码如下：

Algorithm 4: Find-Order

Data: A, B

$n \leftarrow A.length$;
$order = \emptyset$;
for $i \leftarrow 1$ *to* n **do**
  if *IS-SATISFY(order* $\cup (a_i, b_i)$*)* **then**
    $order \leftarrow order \cup (a_i, b_i)$;
end
return $order$;
`IS-SATISFY(` S`)`
{
  $S \leftarrow SortB(S, 2)$;
  $n \leftarrow S.length$;
  $A \leftarrow S[1]$;
  $B \leftarrow S[2]$;
  for $i \leftarrow 1$ *to* n **do**
    if $B[i] < \sum_{j=1}^{i} A[j]$*)* **then**
      return False;
  end
  return True;
}

判断是否为满足订单时时间复杂度为 $O(nlogn)$，故时间复杂度为 $O(n^2logn)$