Lab03-Greedy Strategy

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2021.

∗ If there is any problem, please contact TA Haolin Zhou.
∗ Name:YutianLiu Student ID:519021910548 Email: stau7001@sjtu.edu.cn

1. *Interval Scheduling.* Interval Scheduling is a classic problem solved by **greedy algorithm**: given n jobs and the j-th job starts at s_j and finishes at f_j. Two jobs are compatible if they do not overlap. The goal is to find maximum subset of mutually compatible jobs. Tim wants to solve it by sort the jobs in descending order of s_j. Is this attempt correct? Prove the correctness of such idea, or else provide a counter-example.

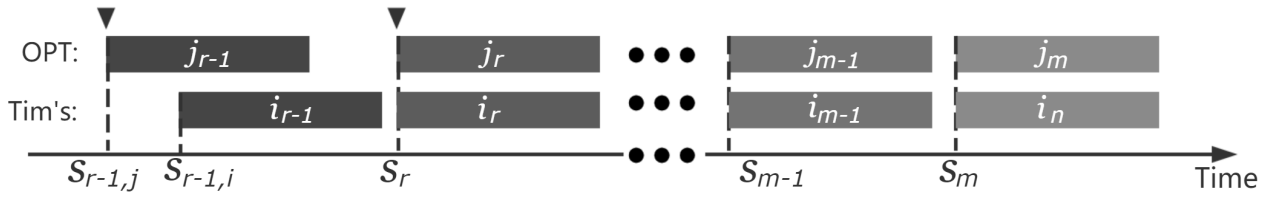
   Solution. Correct.

   Proof. (By contradiction)

   Aussume Tim's method is not optimal.

   Let $i_1, i_2, \cdots, i_n$ donote set of jobs selected by Tim's method.

   Let $j_1, j_2, \cdots, j_m$ donote set of jobs in an optimal solution with $i_n = j_m, i_{n-1} = j_{m-1}, \cdots, i_r = j_r$ for the smallest possible value of r.



   We can see from the chart that job j_{r-1} starts before job i_{r-1}, which means that if we replace job j_{r-1} with i_{r-1}, the solution will still be feasible and optimal, but this contradicts the minimality of r. □

2. *Done deal.* In a basketball league, teams need to complete player trades through matching contracts. Every player is offered a contract. For the sake of simplicity, we assume that the unit is M, and the size of all contracts are integers. The process of contract matching refers to the equation: $\sum_{i \in A} a_i = \sum_{j \in B} b_j$, where a_i refers to the contract value of player i in team A involved in the trade and b_j refers to the value of player j in team B.

   Assume that you are a manager of a basketball team and you want to get **one** star player from another team through trade. The contract of the star player is $n (n \in \mathbb{N}^+)$. The goal is to complete the trade with as few players as possible.

   (a) Describe a **greedy** algorithm to get the deal done with the least players in your team. Assume that there are only 4 types of contracts in your team: $25M$, $10M$, $5M$, $1M$, and there is no limit to the number of players. Prove that your algorithm yields an optimal solution.

   Solution. At each iteration, add the largest contract less than or equal to n, and subtract the value from n.

Algorithm 1: Greedy Algorithm for Manager

Input: An integer n
Output: set of players used in trades S

Sort contracts by value: $c_1 < c_2 < c_3 < c_4$;
$S \leftarrow \emptyset$;
while $n \neq 0$ **do**
    Let k be largest integer s.t. $c_k < n$;
    $n \leftarrow n - c_k$;
    $S \leftarrow S \cup \{c_k\}$;
return S;

□

Proof. (By contradiction)

First, let's start with the preparation for the proof. Any optimal algorithm should have the following 4 properties:

1.The number of players with $1M$ contract for trading cannot exceed 4

If there are $k(k \geq 5)$ $1M$ contracts, we can reduce the players by replacing these contracts with $\lfloor \frac{k}{5} \rfloor$ $5M$ contracts and k mod 5 $1M$ contracts.

2.The number of players with $5M$ contract for trading cannot exceed 1

Similarity, $k(k \geq 2)$ $5M$ contracts can be replaced with $\lfloor \frac{5k}{10} \rfloor$ $10M$ contracts and k mod 2 $5M$ contracts.

3.The number of players with $10M$ contract for trading cannot exceed 2

$k(k \geq 3)$ $10M$ contracts can be replaced with $\lfloor \frac{10k}{30} \rfloor$ $25M$ contracts, $\lfloor \frac{10k}{30} \rfloor$ $5M$ contracts and k mod 3 $10M$ contracts.

4.one player with $5M$ contract and two players with $10M$ contract cannot exist at the same time

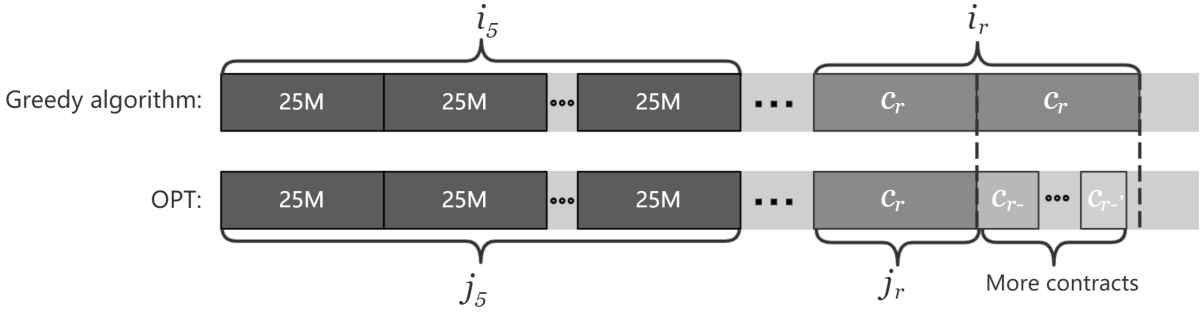
These players can be replaced with only 1 $25M$ contract player.

Aussume this method is not optimal, and there exists another optimal solution.

Let $c_4, c_3, \cdots, c_1$ donote the contract $25M$, $10M$, $5M$, $1M$.

Let $i_4, i_3, \cdots, i_1$ donote the number of player with contract $25M$, $10M$, $5M$, $1M$ selected for trading by this algorithm. And $j_4, j_3, \cdots, j_1$ donote the number of player selected by optimal algorithm with $i_4 = j_4, \cdots, i_r = j_r$ for the largest possible value of $r(r > 1)$.

k	c_k	All optimal solutions must satisfy	Max value of contracts $1, 2, \cdots, k-1$ in any optimal algorithm
1	$1M$	$j_1 \leq 4$	none
2	$5M$	$j_2 \leq 1$	$4M$
3	$10M$	$j_3 \leq 3 \wedge j_2 + j_3 \leq 2$	$4M + 5M = 9M$
4	$25M$	no limit	$10M \times 2 + 4M = 24M$



Since we choose the largest number of maximum contracts each time, $i_r > j_r$.

Case 1: $r = 4, c_r = 25M$. Then the optimal algorithm need to use $10M$, $5M$ and $1M$ to come up with a number greater than or equal to $25M$. But according to the above 4 properties, the the maximum value of the number that can be made up is $1M \times 4 + 10M \times 2 = 24M$. So r can't be 4.

Case 2: $r = 3, c_r = 10$. Then the optimal algorithm need to use$5M$ and $1M$ to come up with a number greater than or equal to $10M$. Similarly, the maximum value is $1M \times 4 + 5M \times 1 = 9M$, so r cannot be 3 either.

Case 3: $r = 2, c_r = 5$. Still similar to the above two cases, the maximum value of the number that can be made up with $1M$ is $1M \times 4 = 4M$, which tells us r cannot be 2.

So $r = 1$, but if $i_4 = j_4, i_3 = j_3, i_2 = j_2$, the remaining is the same, which means $i_1 = j_1$, contradicting our assumptions. □

(b) Suppose that the available contract sizes are powers of c, i.e., the values are $c^0, c^1, \ldots, c^k$ for some integers $c > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.

Proof. (by induction on n)

Similarly, let $i_0, i_1, \cdots, i_k$ donote the number of player with contract $c^0, c^1, \ldots, c^k$ selected for trading by the greedy algorithm. And we have the following properties:

k	c_k	All optimal solutions must satisfy	Max value of contracts $1, 2, \cdots, k-1$ in any optimal algorithm
1	c^0	$i_0 < c - 1$	none
2	c^1	$i_1 < c - 1$	$c^1 - 1$
$\cdots$	$\cdots$	$\cdots$	$\cdots$
k	c^k	$i_k < c - 1$	$c^k - 1$

Consider an optimal way to trade the contract $c_k < n < c_{k+1}$. The greedy algorithm takes contract k, and we'll prove that any optimal solution must also take contract k. If not, it needs to more players with contracts $c_0, \cdots, c_k - 1$ to add up to n. Thus, problem reduces to the trade of contracts $n - c_k$, which, by induction, is optimally solved by greedy algorithm. □

(c) Give a set of contract sizes for which the greedy algorithm does not yield an optimal solution. Your set should include a $1M$ so that there is a solution for every value of n.

Solution. set of contract sizes: $1M$, $10M$, $11M$.

n	Optimal solutions	The greedy algorithm
$20M$	$10M \times 2$	$11M \times 1 + 1M \times 9$
$30M$	$10M \times 3$	$11M \times 2 + 1M \times 8$
$\cdots$	$\cdots$	$\cdots$

In this case, we can modify the greedy algorithm to solve for the optimal strategy

□

Algorithm 2: Greedy Algorithm for Manager_2

Input: contracts value n, contracts array $C[0, 1, \cdots, m]$, current contact cur, Number of contracts cnt (res is initialized to n, cur is initialized to m, cnt is initialized to 0)

Output: number of players used in trades res

```
Sort contracts by value: c_1 < c_2 < ··· < c_m;
if cur < 0 then
    return;
if n%C[cur] == 0 then
    res ← min(res, n/C[cur]);
    return;
for i ← ⌊n/C[cur]⌋ to 0 do
    if cnt + i + 1 >= res then
        return;
    GreedyAlgorithmforManager_2(C[ ], n − C[cur] × i, cur − 1, cnt + i);
return;
```

3. *Set Cover.* **Set Cover** is a typical kind of problems that can be solved by greedy strategy. One version is that: Given n points on a straight line, denoted as $\{x_i\}_{i=1}^{n}$, and we intend to use minimum number of closed intervals with fixed length k to cover these n points.

   (a) Please design an algorithm based on **greedy** strategy to solve the above problem, in the form of *pseudo code*. Then please analyze its *worst-case* complexity.

   Solution. the main idea of this greedy algorithm is using the rightmost line segment to cover from left to right, while ensuring complete coverage of all points.

Algorithm 3: SetCover

Input: An array $X[1, \cdots, n]$

Output: the minimum number of closed intervals with fixed length k

```
sort(X[1, ··· , n]);
start[1] ← X[1];
j ← 1;
for i ← 2 to n do
    if X[i] − start[j] > k then
        j ← j + 1;
        start[j] = X[i];
return j;
```

Worst-case complexity:

Time complexity: $O(n^2)$(using quicksort), $O(n \log n)$(using mergesort).

Since the time complexity of *for* loop is $O(n)$, the *worst-case* time complexity of this algorithm will be determined by the time complexity of the sorting algorithm.

Space complexity: $O(n)$(keeping the starting points), $O(1)$(discarding the starting points)

□

(b) Please prove the correctness of your algorithm.

Proof. (by contradiction)

Aussume greedy algorithm is not optimal, and there exists a optimal algorithm.

Let $i_1, i_2, \cdots, i_n$ donote the starting point of each line from left to right selected by greedy algorithm.

Let $j_1, j_2, \cdots, j_m$ donote the starting point of each line from left to right selected by optimal algorithm with $i_n = j_m, i_{n-1} = j_{m-1}, \cdots, i_r = j_r$ for the largest possible value of r.

We can see from the chart that starting point of closed interval j_{r+1} is before closed interval i_{r+1}, which means that if we replace j_{r+1} with i_{r+1}, the solution will still be feasible and optimal, but this contradicts the maximality of r. □

(c) Please complete the provided source code by C/C++ (The source code *Code-SetCover.cpp* is attached on the course webpage), and please write down the output result by testing the following inputs:

i. the number of points $n = 7$;

ii. the coordinates of points $x = \{1, 2, 3, 4, 5, 6, -2\}$;

iii. the length of intervals $k = 3$.

Remark: Screenshots of running results are also acceptable

Solution. The output is 3, and please see the code in the attachment. □

Remark: You need to include your .pdf and .tex files in your uploaded .rar or .zip file.