

# Lab03-Greedy Strategy

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

\* If there is any problem, please contact TA Haolin Zhou.

\* Name:BeichenYu Student ID:519030910245 Email:polarisybc@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.** This idea is correct. In the common greedy strategy, we sort the jobs in ascending order of  $f_j$ . In fact, the idea of Tim is as same as the common strategy. In this problem we don't care the direction of time, and the only thing we care is the collision. If we let time go upstream, the two strategies will transform each other.

Assume the idea of Tim is not optimal.

Let  $i_1, i_2, \dots, i_k$  denote set of jobs selected by the idea of Tim.

Let  $j_1, j_2, \dots, j_r$  denote set of jobs in an optimal solution with  $i_k = j_k, i_{k-1} = j_{k-1}, \dots, i_r = j_r$  for the smallest possible value of  $r$ .

So  $i_{r-1}$  starts later than  $j_{r-1}$ . However, if we replace  $j_{r-1}$  with  $i_{r-1}$ , solution is still feasible and optimal because interval scheduled before in the optimal solution can also be scheduled although the exchange happened. And that contradicts the minimality of  $r$ .

So the assumption doesn't hold, and the idea of Tim is optimal.  $\square$

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.
- (b) Suppose that the available contract sizes are powers of  $c$ , i.e., the values are  $c^0, c^1, \dots, c^k$  for some integers  $c > 1$  and  $k \geq 1$ . Show that the greedy algorithm always yields an optimal solution.
- (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.** (a) First choose as many  $25M$  contacts as possible. When the remain contacts is less than  $25M$ , choose as many  $10M$  contacts as possible. When the remain contacts

is less than  $10M$ , choose as many  $5M$  contracts as possible. When the remain contacts is less than  $5M$ , use enough  $1M$  contracts to fill up.

First let us assume that if using this greedy algorithm,  $n = m_1 \times 1M + m_2 \times 5M + m_3 \times 10M + m_4 \times 25M$ .  $m_1 < 5$  is obvious because if we have 5 players with a  $1M$  contract, a player with a  $5M$  contract is a better choice. For the same result,  $m_2 < 2$ , and 2 of  $10M$  player can not exist together with a  $5M$  player. We proof that the number of  $25M$  must be  $m_3$ . We assume that there is a better algorithm and  $n = p_1 \times 1M + p_2 \times 5M + p_3 \times 10M + p_4 \times 25M$ . Because we use the greedy algorithm,  $m_3 \geq p_3$ . But if  $p_3 < m_3$ , there is at least  $25M$  must use  $1M, 5M$  and  $10M$  to get. But it is impossible because we have at most 4 of  $1M$  players, 1 of  $5M$  player and available  $10M$  players. So  $m_3 = p_3$ . Using the same method, we can proof that  $m_2 = p_2, m_1 = p_1$  and  $m_0 = p_0$ . That is to say that the greedy algorithm is right.

- (b) If using the greedy algorithm,  $n = m_0c^0 + m_1c^1 + \dots + m_kc^k$  and we assume that there is a better algorithm,  $n = n_0c^0 + n_1c^1 + \dots + n_kc^k$ .

First, we proof that  $n_k = m_k$ . As it is a greedy algorithm,  $m_k \geq n_k$ . If  $n_k < m_k$ , then  $n_k \leq m_k - 1$ . It is absolutely that  $n_0, n_1, \dots, n_{k-1} < c$  as if there is  $c$  contracts they can be changed to a bigger contract. So  $n_0c^0 + n_1c^1 + \dots + n_{k-1}c^{k-1} \leq (c-1)c^0 + (c-1)c^1 + \dots + (c-1)c^{k-1} = c^k - 1 < c^k$ . So  $n = n_0c^0 + n_1c^1 + \dots + n_kc^k < c^k + n_kc^k \leq m_kc^k < n$ , and it is absolutely wrong. So  $n_k = m_k$ .

Using the same method, we can proof that for  $\forall i \leq k, n_i = m_i$ . That is to say that the greedy algorithm is correct.

- (c) Assume that there are only 3 types of contracts in your team:  $8M, 5M, 1M$ , and  $n = 20M$ . If using the greedy algorithm, the solution should be  $8M + 4 \times 1M$ , 6 players in total. This is not the optimal obviously as  $4 \times 5M$  can solve it as well.

□

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.

- 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.
- Please prove the correctness of your algorithm.
- Please complete the provided source code by C/C++, and please write down the output result by testing the following inputs:
  - the number of points  $n = 7$ ;
  - the coordinates of points  $x = \{1, 2, 3, 4, 5, 6, -2\}$ ;
  - the length of intervals  $k = 3$ .

**Remark:** Screenshots of running results are also acceptable

**Solution.** (a) The pseudo code is as follows.

---

**Algorithm 1:** Set Cover

---

```
1 Sort the coordinates of points so that  $x_1 \leq x_2 \leq \dots \leq x_n$ ;
2  $ans \leftarrow 1$ ;  $index \leftarrow 1$ ;  $left \leftarrow x_1$ ;  $right \leftarrow left + k$ ;
3 while  $index < n$  do
4    $index \leftarrow index + 1$ ;
5   if  $x[index] > right$  then
6      $ans \leftarrow ans + 1$ ;
7      $left \leftarrow x[index]$ ;  $right \leftarrow left + k$ ;
8 return  $ans$ ;
```

---

In the sorting part, the time complexity in the worst case is  $O(n^2)$ . And when doing the loop the time complexity is always  $O(n)$ , so finally the time complexity in the worst case is  $O(n^2)$ .

(b) Assume the idea of Tim is not optimal.

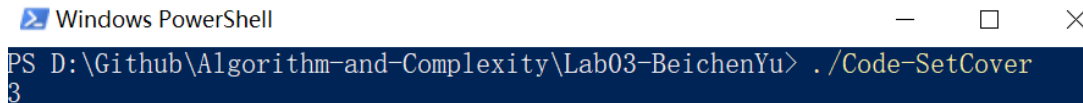
Let  $l_1, l_2, \dots, l_k$  denote intervals selected by the greedy algorithm.

Let  $s_1, s_2, \dots, s_k$  denote intervals selected by an optimal solution with  $l_1 = s_1, l_2 = s_2, \dots, l_r = s_r$  for the biggest possible value of  $r$ .

So the left endpoint of  $s_{r+1}$  is smaller than  $l_{r+1}$ . However, if we replace  $s_{r+1}$  with  $l_{r+1}$ , solution is still feasible and optimal because the points after the interval can still be covered although the exchange happened. And that contradicts the maximality of  $r$ .

So the assumption doesn't hold, and the greedy algorithm is optimal.

(c) I have finish it and the source code is included in the zip. The output is 3, and the screenshot of running results is below.



```
Windows PowerShell
PS D:\Github\Algorithm-and-Complexity\Lab03-BeichenYu> ./Code-SetCover
3
```

Figure 1: The screenshot of running results

□

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