Lab03-Greedy Strategy

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

∗ If there is any problem, please contact TA Haolin Zhou.
∗ Name:Xin Xu Student ID:519021910726 Email: xuxin20010203@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. No. This attempt isn't correct for the job with an earlier starting time may have a later finishing time, too. The counter-example is below. □

Figure 1: The Counter Example

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 $c^0, c^1, \ldots, c^k$ 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) We can apply a greedy algorithm recursively. Suppose the value of the star player is x and the 4 types of contracts in an increasing order are c_1, c_2, c_3, c_4. For every $c_k \leqslant x < c_{k+1}$, we choose the contract with value of c_k and recursively apply this method to $x - c_k$ untill the remainder is zero.
   We will prove this algorithm is optimal by contradiction. If the greedy algorithm isn't optimal, and there is another optimal method, which quiz to choose c_k for the condition that $c_k \leqslant x < c_{k+1}$. In this case, it must choose $c_1, c_2, ..., c_{k-1}$ to meet the value of c_k. If $c_k = 25M$, there are at least three contracts with values of $\{10M, 10M, 5M\}$. If $c_k = 10M$, there are at least two contracts with values of $\{5M, 5M\}$. If $c_k = 5M$,

there are at least five contracts with values of $\{1M, 1M, 1M, 1M, 1M\}$. Considering all conditions, the number of contracts are all larger than one. And we can figure out a more optimal method by replacing the chosen contracts with the one that greedy algorithm picks, which is contract to our hypothesis. So, the greedy algorithm is optimal.

(b) We can easily get the idea that in an optimal method, the number of all the contracts is less than c except the one with largest value because for any number $\geqslant c$, we can replace c contracts of that value with just one contract of the higher value. So, we can prove it by contradition. If we don't choose c^k in the situation that $c^k \leqslant x < c^{k+1}$, we can only choose $(c-1)c^{k-1} + (c-1)c^{k-2} + ... + (c-1)c^2 + (c-1)c^1$, whose sum is just $c^k - 1$. To meet up the value of c^k, we must choose one more contract, which is contrast to our observation. So, the greedy algorithm is an optimal solution in this case.

(c) The set of contract sizes are $\{50M, 35M, 10M, 1M\}$. In this case, if we want to get a contract with the value of $70M$, greedy algorithm picks $\{50M, 10M, 10M\}$ while the optimal method is to pick $\{35M, 35M\}$.

□

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.

(b) Please prove the correctness of your algorithm.

(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. (a) The pseudo code is below.

Algorithm 1: Greedy

Input: An array $x[1, \cdots, n]$, the number of points n, and the length of intervals k.
Output: The minimum number of closed intervals with fixed length k to cover these n points.

```
Sort n points by values so that x_1 < x_2 < ... < x_{n-1} < x_n;
i ← 2; num ← 1; start ← x_1; end ← start + k;
while end < x_n do
    while end ⩾ x_i do
        ++i;
    start ← x_i;
    end ← x_i + k;
    ++num;
return num;
```

Time Complexity. In worst case, the time complexity of sorting is $O(n^2)$ and the time complexity of the while-loop is $O(n)$. So, the time complexity of worst case is $O(n^2)$.

(b) **Proof.** We will prove the greedy algorithm by contraction. In the greedy algorithm, each interval starts at an exact point in such as x_i. And if an interval starts before x_i but doesn't reach the point x_{i-1}, we can move it to x_i which doesn't decrease the number of points that interval covers, but may increase it. So, suppose there is an another optimal method with the max number of intervals starting exactly at x_i, which we assume as r. We can replace exceptional intervals with ones creating by greedy algorithm. And this action leads to at least as optimal method as former one, which is contrast to our hypothesis that the max number is r. So, the greedy algorithm is optimal. □

(c) The .cpp file is in the folder, and the result is 3.

□

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