Lab03-Greedy Strategy

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

∗ Name: Haoyi You Student ID: 519030910193 Email: yuri-you@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. Such idea is correct.Every time he only need to choose the *max* f_j in the jobs that are compatible with the jobs chosen.
Proof: Assume $\mathbb{A}$ is a random given job set. $\mathbb{B}$ is the chosen job set using the algorithm above and $\mathbb{C}$ is one of the best solution job set with maximum $|\mathbb{B} \cap \mathbb{C}|$.
Let b be the element that has minimum s_b in $\mathbb{B} - \mathbb{C}$ and c be the element that has minimum s_j in $\mathbb{C} - \mathbb{B}$.
Let $\mathbb{D}$ be subset of $\mathbb{B} \cap \mathbb{C}$ and every element in $\mathbb{D}$ has larger s_j than s_b. And $\mathbb{E} = \mathbb{C} - \mathbb{D} - \{b\}$. It is obvious that b or c is compatible with, and c is also compatible with $\mathbb{E}$, which means $\forall e \in \mathbb{E}$,$f_e \leq s_c$. Since $s_b \geq s_c$, b is also compatible with $\mathbb{C}$, such that we change c to b in $\mathbb{C}$ getting $\mathbb{C}'$. $\mathbb{C}'$ is also a best solution, which is against the assume.

□

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)

Algorithm:
Choose the max number of $25M$ contracts and for the remain number, choose the max number of $10M$ contracts and so on.
Proof:
Assume my algorithm get the number of each contracts are x_1, x_2, x_3, x_4 (from $25M$

to $1M$). And there is a more optimal solution with y_1, y_2, y_3, y_4. Mark result $Y = y_1 + y_2 + y_3 + y_4$. If $y_2 \geq 3$,let $y_2 - 3, y_1 + 1, y_3 + 1$, the whole money does not change but Y decreases.
Similarly, if $y_3 \geq 2$ let $y_3 - 2, y_2 + 1$. If $y_4 \geq 5$,let $y_3 + 1, y_4 - 5$.
As a result, y_1, y_2, y_3, y_4 all satisfy the greedy algorithm above, which means $(x_1, x_2, x_3, x_4) = (y_1, y_2, y_3, y_4)$.

(b) Proof:
Assume in the optimal solution,the number of each contract is d_i.
Assumption: $\forall c$, if $\exists j \leq k-1$ with $d_j \geq c$. We can let $d_j - c, d_{j+1} + 1$ get a new sequence of $\{d_i^{'}\}$.

$$\begin{gathered}\sum_{i=1}^{k} d_i * c^i = \sum_{i=1}^{k} d_i^{'} * c^i = n \\ \sum_{i=1}^{k} d_i - d_i^{'} = c - 1\end{gathered} \tag{1}$$

That conflicts with the assumption. So $\forall c, \forall j \leq k-1, d_j < c$. So the optimal solution is greedy algorithm.

(c) The values are $25M, 21M, 10M, 1M$, and the sum $n = 63M$. If we use the greedy algorithm, we get

$$63M = 2 * 25M + 10M + 3 * 1M \tag{2}$$

But we can also use

$$63M = 3 * 21M \tag{3}$$

In method1 the number of contracts is 6 while in method2 is 3.

□

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)

```
Algorithm 1: Set Cover
  Input: An array $A[x_1, \cdots, x_n]$,the length $K$
  Output: The minimum number t
1 sort($A[x_1, \cdots, x_n]$);
2 $t \leftarrow 0$ $i \leftarrow 0$;
3 while $i < n$ do
4     $final \leftarrow A[i] + K$;
5     while $i < n$ && $A[i] < final$ do
6         $i \leftarrow i + 1$;
7     $t \leftarrow t + 1$;
8 return $t$
```

The complexity of worst-case is $O(nlogn)$.
No matter what situation it is,the sort cost $O(nlogn)$(regardless of bucket sorting).And the loop at most run $O(n)$(i from 0 to $n-1$).So the complexity is $O(nlogn)$ in every case.

(b) Proof:
In this algorithm, assume the j^{th} time program enters the outer loop when $i = a_j$. So the chosen intervals are $\{[x_{a_j}, x_{a_j} + K]\} = \mathbb{X}$. Assume the optimal solution of intervals are $\mathbb{Y}$. We can assume each two intervals in $\mathbb{Y}$ are disjointed and sorted. (If $[y_i, y_i + K], [y_j, y_j + K], y_i < y_j$ are not disjointed, we can change the $[y_i, y_i + K]$ to $[y_j - K, y_j]$,and it covers more space.)
Assume i is the smallest number satisfies $[x_{a_i}, x_{a_i} + K] \neq [y_i, y_i + K]$.
If $y_i > x_{a_i}$, since each intervals in $\mathbb{Y}$ is disjointed, the x_{a_i} is not in any intervals in $\mathbb{Y}$, which is impossible.
If $y_i < x_{a_i}$, we can change the $[y_i, y_i + K]$ to $[x_{a_i}, x_{a_i} + K]$ in $\mathbb{Y}$ (adjust other intervals after this interval), it also satisfies the condition,and $\mathbb{Y}'$ is also a optimal solution. And this operations continues until $i == t$.This time $\mathbb{X} = \mathbb{Y}$

(c) The code is in the appendix.

□

A Appendix

```
#include <iostream>
using namespace std;
void quickSort(int s[], int l, int r)
{
    if (l< r)
    {
        int i = l, j = r, x = s[l];
        while (i < j)
        {
            while(i < j && s[j]>= x)
```

```
                j--;
            if(i < j)
                s[i++] = s[j];
            while(i < j && s[i]< x)
                i++;
            if(i < j)
                s[j--] = s[i];
        }
        s[i] = x;
        quickSort(s, l, i - 1);
        quickSort(s, i + 1, r);
    }
}
int Greedy(int x[], int k, int n)
{
    quickSort(x, 0, n-1);
    int t = 0, i = 0;
    while (i < n) {
        int destination = x[i] + k;
        while (i < n && x[i] < destination)++i;
        ++t;
    }
    return t;
}
int main()
{
    int x[7]={1,2,3,4,5,6,-2};
    int k=3;
    int n=sizeof(x) / sizeof(x[0]);
    int num_interval=Greedy(x,k,n);
    cout << num_interval << endl;
    return 0;
}
```