Lab03-Greedy Strategy

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2018.

∗ If there is any problem, please contact TA Xinyu Wu.
∗ Name: Hongyi Guo Student ID: 516030910306 Email: guohongyi@sjtu.edu.cn

1. **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 pseudo code of the algorithm is as follows.

   Algorithm 1: Set Cover

   input : n, k, $\{x_i\}_{i=1}^n$.
   output: Minimum number of closed intervals with fixed length k.

   ```
   Mergesort {x_i}_{i=1}^n in an ascending order;
   result ← 1;
   prev ← x_1;
   for i ← 1 to n do
       if x_i − prev > k then
           // The last interval cannot cover it.
           // We let a new interval begin at it.
           result ← result + 1;
           prev ← x_i;
   output result;
   ```

   Analysis of worst case.

   The worst complexity of mergesort is $O(n \log n)$.

   The **for** loop will be executed n times. The complexity of each iteration is just $O(1)$. So the complexity of **for** loop is $O(n)$.

   Thus, the worst time complexity of the algorithm should be $O(n \log n)$. □

   (b) Please prove the correctness of your algorithm.

   Solution. We assume the intervals are put one by one and their coordinates are in the ascending order. That's why we need to sort the points in the ascending order. So the first interval must cover the first point.

   The first interval should begin where the first point locates. The reason is quite obvious. If it starts at a position forward, it will cover the empty space on the left of the first points where there is no points. That is a waste of interval. But if it starts a little backward, it will not cover the first point, which contradicts that it's the first interval. Thus, the first interval begins where the first point locates. And for the same reason, the second interval begins where locates the first point that the last interval cannot cover. And so does the third, forth... points. So we've proved the correctness of the greedy 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 $k = 3$.

```
int Greedy(int x[], int k, int n)
{
    quickSort(x, 0, n-1);
    int result = 1, prev = x[0];
    for (int i = 0; i < n; i++) {
        if (x[i] - prev > k) {
            result++;
            prev = x[i];
        }
    }
    return result;
}
```

2. G is an undirected graph. A set of cycles $\{c_1, c_2, \ldots, c_k\}$ in G is called redundant if every edge appearing in at least one cycle in $\{c_1, c_2, \ldots, c_k\}$ appears in an even number of c_i's. A set of cycles is *independent* if it contains no redundant subset. A maximal independent set of cycles is called a *cycle basis* for G.

(a) Let $\mathbf{C}$ be any cycle basis for G. Prove that for any cycle γ in G, there is a subset $A \subseteq \mathbf{C}$ such that $A \cup \{\gamma\}$ is redundant. In other words, γ is "exclusive" of the cycles in A.

Proof. Assume there is no subset $A \subseteq \mathbf{C}$ such that $A \cup \{\gamma\}$ is redundant. We take $C_1, C_2, \ldots, C_n$ as the subsets of $\mathbf{C}$. Then none of them is redundant. Then we let $\mathbb{C}' = C \cup \{\gamma\}$. All subsets of $\mathbb{C}'$ are $C_1, C_2, \ldots, C_n, C_1 \cup \{\gamma\}, C_2 \cup \{\gamma\}, \ldots, C_n \cup \{\gamma\}, \{\gamma\}$. $C_1, C_2, \ldots, C_n$ are obvious not redundant. $C_1 \cup \{\gamma\}, C_2 \cup \{\gamma\}, \ldots, C_n \cup \{\gamma\}$ are not redundant because of the assuming. $\{\gamma\}$ is not redundant for it only contains one cycle. Thus, none of those subsets is redundant. So $\mathbb{C}'$ is an independent set with more cycles than $\mathbf{C}$, which contradicts with that $\mathbf{C}$ is a maximal independent set of cycles.

□

(b) Prove that the set of independent cycle sets form a matroid.

Proof.

First, we prove that the set of independent cycle $\mathbf{C}$ satisfies the hereditary property by proving $A \subset B$, $B \in \mathbf{C} \Rightarrow A \in \mathbf{C}$. If $A \subset B$, $B \in \mathbf{C}$, then none of the subsets of B is redundant. Considering all the subsets of A should also be subsets of B, they are not redundant, either. So, A is independent, $A \in \mathbf{C}$.

Next, we prove that $\mathbf{C}$ satisfies the exchange property by proving $A, B \in \mathbf{C}$ and $|A| < |B|$ $\Rightarrow \exists x \in B \backslash A \ s.t. \ A \cup \{x\} \in \mathbf{C}$.

Assume there are totally m edges in G, $e_1, e_2, \ldots, e_m$. We make a function f, whose input is a set of cycles S and output is a m-length row vector of 0 and 1. The i_{th} element indicates whether the i_{th} edge appears in an even(0) or odd(1) number of S_i's. Obviously $f(S_1 \cup S_2) = f(S_1) + f(S_2) \pmod 2$ $(S_1 \cap S_2 = \emptyset)$. We put some rows A_i of A into a set

S, the corresponding output f of them is $\sum_i f(A'_i) \pmod 2$. So the operation of uniting A_i's equals to do linearly transformation to A'_i's. A set of cycle is redundant when and only when its output f is all zero vector. Meanwhile, $f(\{A_i\}), f(\{A_i\}), \ldots, f(\{A_a\})$ is linearly dependent if and only if after a series of linearly transformation, some of them becomes zero vector.

So, A is independent if and only if every row of A' is linearly independent.

Because $A_1, A_2, \ldots, A_a$ is linearly independent, every B_i is dependent with at most one A_i. For arbitrary B_i and B_j, they cannot be dependent with the same A_k because $B_1, B_2, \ldots, B_b$ is independent. Thus, at most a vectors of B is dependent with A. So for $a < b$, there must be an i such that $A_1, A_2, \ldots, A_a, B_i$ are independent. Thus, $\mathbf{C}$ satisfies the exchange property.

Together, $\mathbf{C}$ is matroid.

□

(c) Now suppose each edge of G has a weight, and G contains n vertices and m cycles. Define the weight of a cycle to be the total weight of its edges, and the weight of a set of cycles to be the total weight of all cycles in the set. (Thus, each edge is counted once for every cycle in which it appears.) Design an efficient algorithm to compute the minimum-weight cycle basis in G in the form of *pseudo code*, and analyze its *worst-case* time complexity.

Solution. The pseudo code:

Algorithm 2: Minimum-weight Cycle Basis

input : n vertices, m cycles, l edges; cycles are given by array of vectors mentioned in the 2nd question.
output: The minimum-weight cycle basis.

```
1  Calculate the weight of every cycle;
2  Sort the cycles based on their weight in an descending order;
3  C is a m × l all-zero matrix;
4  Results ← ∅;
5  cnt ← 0;
6  for i ← 1 to m do
7  |  C' ← C;
8  |  Add cycles_i to C';
9  |  Use Gaussian Elimination to calculate the rank r of C';
10 |  if r = cnt + 1 then
11 |  |  C ← C';
12 |  |  Add cycles_i into Results;
13 output Results;
```

Worst-case time complexity analysis.

The worst-case of time complexity of the sort is $O(m \log m)$ if we use mergesort. The **for** loop is executed m times. The worst time complexity of copying Matrix C is $O(ml)$. The Gaussian Elimination is at most $O(m^3)$. Thus, the worst time complexity of for loop is $O(m^3)$.

In conclusion, the worst-case time complexity is $O(m^3)$. □

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