Lab03-Greedy Strategy

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

∗ If there is any problem, please contact TA Xinyu Wu.
∗ Name:Juncheng Wan Student ID:516021910620 Email: 578177149@qq.com

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. Pseudo code is as follows:

   Algorithm 1: Greedy-SetCover

   Input: A set S of n points on a straight line; the fixed length k of closed intervals.
   Output: the minimum number t of closed intervals

   ```
   In the number axis, sort all elements in S into ordering x_1 ≤ x_2 ≤ ··· ≤ x_n ;
   pos ← x_1 ;
   t ← 1 ;
   for i = 2 to n do
       if x_i ≤ pos + k then
           continue ;
       else
           pos ← x_i ;
           t ← t + 1 ;
   return t
   ```

   If $x_2 - x_1 > k, x_3 - x_2 > k, \cdots, x_n - x_{n-1} > k$, we have the *worst-case*. The time complexity of the **for** loop is $O(n)$, because no matter how we have to compare $O(n)$ times. The time complexity of sorting is $O(nlogn)$. Therefore, the time complexity of Greedy-SetCover $T(n) = O(nlogn + n) = O(nlogn)$. □

   (b) Please prove the correctness of your algorithm.

   Proof. Assume greedy is not optimal, and let's see what happens.
   Let $[a_1^g, b_1^g], [a_2^g, b_2^g], \cdots, [a_k^g, b_k^g]$ denote intervals selected by greedy, where $a_1^g < a_2^g < \cdots < a_k^g$.
   Let $[a_1^*, b_1^*], [a_2^*, b_2^*], \cdots, [a_m^*, b_m^*]$ denote intervals in the optimal solution, where $a_1^* < a_2^* < \cdots < a_m^*$.
   Set $a_1^g = a_1^*, a_2^g = a_2^*, \cdots, a_r^g = a_r^*$ for the largest possible value of r. If $a_1^g \neq a_1$, set $r = 0$.
   According to the Greedy-SetCover algorithm, $a_{r+1}^g = x_i \in S$. Because the optimal solution should cover x_i, $a_{r+1}^* < x_i$. If $a_{r+1}^* = a_{r+1}^g$, $[a_{r+1}^g, b_{r+1}^*]$ will also cover x_i and the other points covered in $[a_{r+1}^*, b_{r+1}^*]$. Therefore, it is also an optimal solution. However it contradicts 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 $k = 3$.

Solution. The output result is 3. □

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 in G 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 that for any cycle γ in G, for any $A \subseteq \mathbf{C}$, $A \cup \{\gamma\}$ is not redundant. First, $\{\gamma\}$ itself is not redundant. Set $B = \mathbf{C} \cup \{\gamma\}$. The subsets of B contain $\{\gamma\}$, any $A \cup \{\gamma\}$ (where $A \subsetneq \mathbf{C}$). Therefore, B contains no redundant subset and B is independent. It contradicts the maximality of $\mathbf{C}$.

□

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

Proof.

Hereditary: If $\mathbf{C}$ is a set of independent cycle sets, all subsets of $\mathbf{C}$ is not redundant. For any $A \subseteq \mathbf{C}$, all the subsets of A are the subsets of $\mathbf{C}$. Therefore, all subsets of A is not redundant. A is a set of independent cycle sets.

Exchange Property: Consider $A, B \subseteq \mathbf{C}$ with $|A| < |B|$. If $\forall x \in B \backslash A$, $A \bigcup \{x\} \nsubseteq \mathbf{C}$. See what happens.

Set $|A \cap B| = t, |A - A \cap B| = n, |B - A \cap B| = m$. Since $|A| < |B|$, get $n < m$. Since $\forall x \in B \backslash A$, $A \bigcup \{x\} \nsubseteq \mathbf{C}$, $\forall x \in B \backslash A$, $\exists A_x \subseteq A, s.t. A_x \cup x$ is redundant.

It is impossible that for all cycles in A_x belong to $A \cap B$. If so, as B is independent, $A_x \cup x \subseteq B$ is not redundant. Therefore, there must be a cycle in A_x belonging to $A - A \cap B$.

It is obvious that cycles in B is different from each other. Because B is a independent system, if there are two cycles c_i, c_j in B are the same, the subset $\{c_i, c_j\} \subseteq B$ must be redundant. Therefore all cycles $\{x_1, x_2, \cdots, x_m\}$ in $B - A \cup B$ are different from each other.

Set E_{A_x} a set of all edges in A_x and having **odd** number in G. Set E_x a set of all edges in cycle x. Because $A_x \cup x$ is redundant, $E_{A_x} = E_x$. I get $E_{A_{x_i}} = E_{x_i}, \forall i = 1, 2, \cdots, m$.

Since all cycles $\{x_1, x_2, \cdots, x_m\}$ in $B - A \cup B$ are different from each other, $\{E_{x_1}, E_{x_2}, \cdots, E_{x_m}\}$ are different from each other.

Select $x_i, x_j \in \{x_1, x_2, \cdots, x_m\}$ and $x_i \neq x_j$. Therefore, $E_{A_{x_i}} \neq E_{A_{x_j}}$. To form each odd edge in $E_{A_{x_i}}$ or $E_{A_{x_j}}$, there should be at least two cycles in A_{x_i} and A_{x_j} that are different and these two cycles belonging to $A_{x_i} - A_{x_i} \cap B$ and $A_{x_j} - A_{x_j} \cap B$.

Because all m cycles in $B - A \cap B$ are different, there should be at least m different cycles in $A - A \cap B$. Therefore, $n \geq m$, which contracts with $n < m$. □

(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. Pseudo code is as follows:

Algorithm 2: Greedy-MinimunWeightCycleBasis

Input: The weight of each edge in graph G
Output: the minimum-weight cycle basis in G

```
Find all cycles in G;
Sort all weights of cycles in G into ordering w(c_1) ≥ w(c_2) ≥ ··· ≥ w(c_m) ;
A ← ∅ ;
for i = 1 to m do
    Use recursive function to find all the subsets of A, A_1, A_2, ··· , A_{n_i} ;
    flag ← true ;
    for j = 1 to n_i do
        if A_j ∪ c_i is redundant then
            flag ← false ;
            break ;
    if flag = true then
        A ← A ∪ c_i ;
return A
```

Set the time complexity of Greedy-MinimumWeightCycleBasis $T(m)$, where m is the number of edges in G. In the worst case, the time complexity of sorting is $O(m^2)$.

In the outer **for** loop, the time complexity of finding all the subsets of A is $O(2^m)$. Because if the set of all the circles in G are independent， the size of A will be m and the number of subsets of A will be 2^m.

In the inner **for** loop, set the time complexity of judging if a set is redundant $O(1)$. As the maximum n_i is 2^m, calculate $T(m)$ as follows:

$$\begin{aligned} T(m) &= O(m^2) + O(m(2^m + 2^m)) \\ &= O(m^2 + m2^m) \\ &= O(m2^m) \end{aligned}$$

□

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