Algorithms Homework #4

Due: 2017/12/27 (Wed.) 03:00 (Programming)
2017/12/27 (Wed.) 14:20 (Hand-written)

Contact TAs: alg2017ta@gmail.com

Instructions

- There are two sections in this homework, including the hand-written part and the programming part.
- **Hand-written.** Please submit your answers to the instructor at the beginning of the class. TA will collect the answer sheets in the first break of the class. **NO LATE SUBMISSION IS ALLOWED**.
- **Programming.** Please upload your source codes in a single .tar or .zip file (e.g., b04901001.zip) to Ceiba. You must implement your program with **Python 3**.
- **Delay policy.** You have a six-hour delay quota for programming assignments for the whole semester. Once the quota is used up, any late submission will receive no credits.
- **Collaboration policy.** Discussion with other students is strongly encouraged, but you must obtain and write the final solution by yourself. Please specify the references (e.g., the name and student id of your collaborators and/or the Internet URL you consult with) for each **hand-written and programming question** on your answer sheet of the hand-written problems. If you solve some problems by yourself, please also specify "no collaborator". Homeworks without reference specification may not be graded.
- **Academic honesty.** Cheating is not allowed and is against university policy. Plagiarism is a form of cheating. If cheating is discovered, all students involved will receive an F grade for the course (NOT negotiable).

Hand-Written Problems

Problem 1 (15%)

Please answer the following questions:

1. (5%) Suppose we perform a sequence of n operations on a data structure in which the i-th operation costs i if i is an exact power of 2, and 1 otherwise. Use aggregate analysis to determine the amortized cost per operation.

2. (5%) Suppose we perform a sequence of stack operations, i.e., PUSH and POP, on a stack whose size never exceeds k. After every k operations, we make a copy of the entire stack for backup purposes. Show that the cost of n stack operations, including copying the stack, is $O(n)$ by assigning suitable amortized costs to the various stack operations.

3. (5%) Consider an ordinary **Binary Min-Heap** data structure with n elements supporting the operations of INSERT and EXTRACT-MIN in $O(\lg n)$ worst-case time. Give a potential function Φ such that the amortized cost of INSERT is $O(\lg n)$ and the amortized cost of EXTRACT-MIN is $O(1)$, and show that it works.

Problem 2 (20%)

Since we do not always know in advance how many objects some applications will store in a table, we might allocate space for a table, only to find out later that it is not enough. We must then reallocate the table with a larger size and copy all objects stored in the original table over into the new, larger table. That's how dynamic tables work. Let's define load factor α to be the number of items stored in the table divided by the size (number of slots) of the table.

1. (10%) Suppose our dynamic table only supports INSERT operation, that is, our table will only be expanded. If the table is full before insertion of an item, we expand the table by doubling its size. By defining a suitable potential function Φ, please show that the amortized cost for the INSERT operation is $O(1)$. (*Hint*: you need to consider different cases of α when you do the operation.)

2. (10%) Suppose our dynamic table now supports both INSERT and DELETE operations, that is, our table now can be expanded or be contracted. If the table is full before insertion of an item, we expand it by doubling its size. If the table is below (1/4)-full after deletion, we contract it by halving its size. Now we have to define suitable potential functions Φ for case of $\alpha \geq 0.5$ and $\alpha < 0.5$, respectively. Please show that the amortized cost for both INSERT and DELETE operations are $O(1)$. (*Hint*: you need to consider different cases of α when you do the operations.)

Problem 3 (20%)

For a sorted array, while it takes logarithmic time to search an element by binary search, it still needs linear time to insert a new element. Frank wants to improve the time for insertion by designing a data structure which keeps several sorted arrays.

Frank decides to design both SEARCH and INSERT operations on a set of n elements. First, let $k = \lceil \lg(n+1) \rceil$ and let the binary representation of n be $\langle n_{k-1}, n_{k-2}, ..., n_0 \rangle$. We have k sorted arrays $A_0, A_1, ..., A_{k-1}$, where for $i = 0, 1, ..., k-1$, the length of array A_i is 2^i. Each array is either full or empty. $n_i = 1$ denotes A_i is full, while $n_i = 0$ denotes A_i is empty. The total number of elements held in all k arrays is $\sum_{i=0}^{k-1} n_i 2^i = n$. While each individual array is sorted, elements in different arrays bear no particular relationship to each other.

1. (10%) Please help Frank perform SEARCH operation and analyze its worst-case running time.
2. (10%) Please help Frank perform INSERT operation and analyze its worst-case amortized running time.

Problem 4 (10%)

There are several cities which are connected by some roads in the country. Everyone can travel from one city to any other city via those roads without restriction. One day, the president would like to expropriate one road in use of holding a Carnival, but he would not like to make any resident feel inconvenient when traveling. So, the goal is to help him determine whether he can block one arbitrary road between two cities without making any impossibilities for arbitrary journey between any two cities.
Please design a **polynomial time algorithm** or prove that this problem is **NP-complete**. You may assume that 3-SAT, Vertex Cover, Hamiltonian Path and Hamiltonian Cycle problems are all NP-complete. Also, you do not need to prove that the problem is in NP.

Problem 5 (10%)

Given a graph $G(V, E)$ in which every edge e has a positive edge cost c_e. Given two vertices u, v and a constant k, the goal is to determine whether there exists a simple path from u to v with total cost at least k.
Please design a **polynomial time algorithm** or prove that this problem is **NP-complete** for general graphs. You may assume that 3-SAT, Vertex Cover, Hamiltonian Path and Hamiltonian Cycle problems are all NP-complete. Also, you do not need to prove that the problem is in NP.

Problem 6 (10%)

Frank has become a world-class wizard after his graduation from Hogwarts. One day, he finds a recipe for lots of ancient Magic Potions in the Room of Requirement, but the recipe has a curse on it. Frank needs to reproduce at least k kinds of potions; otherwise, due to the curse, all potions will explode. He sneaks in Snape's laboratory which contains a bunch of different chemistry components satisfying all ingredients listed on the recipe. **But each component has the limited amount for only one potion.** The goal for Frank is to determine whether he can reproduce any given k kinds of potions or not.
Please design a **polynomial time algorithm** or prove that this problem is **NP-complete**. You may assume that 3-SAT, Vertex Cover, Hamiltonian Path and Hamiltonian Cycle problems are all NP-complete. Also, you do not need to prove that the problem is in NP.

Programming Problems

Note that in the programming problems, input is from the **standard input** and you should output the result to the **standard output**. You **cannot import any module or library**. Also, you can only implement your program with **Python3**. Only `p1.py` should be uploaded. Put the file(s) in a folder named {student id} and compress the folder as .zip or .tar format.

Problem 1 (15%)

With your great help, Frank has successfully produced those ancient Magic Potions. Surprisingly, one of the Magic Potion ***X*** has been found to have the power to drive away the Dementors. Here comes the problem, every city has several powerful wizards protecting it, **but the road between cities don't**. As a result, they need the help of Frank's potion. Luckily, once the city has been sprayed with this Magic Potion (What a relief to those wizards!), the adjacent roads of that city will be protected, too. Therefore, the goal is to help Frank determine the minimum number of cities needed to be sprayed to make all roads be protected from the evil Dementors.

Input Format

The first line contains two integers C and R, indicating the number of cities, and the number of roads, respectively. The following R lines contain two integers i and j, where i indicates the first city (from 1 to V) and j indicates the other city connected with i through a road.

- $1 \leq C \leq 1 \times 10^3$
- $0 \leq R \leq 1 \times 10^5$

Output Format

In the first line, please output the minimum number of cities needed. In the second line, please output the set of cities ids (from small to large and separated by a space).

Time Limit

30 seconds

Grading Policy

- Output any legal solution correctly in the time limit: 15 points (no partial)
- Help Frank choose less than 50% of solutions in our class: bonus 5 points
- Help Frank choose less than 80% of solutions in our class: bonus 5 points

Sample Input 1

```
7 8
1 2
1 4
1 6
2 3
2 5
3 4
3 5
6 7
```

Sample Output 1

```
4
1 2 3 6
```

Sample Input 2

```
5 5
1 2
1 3
2 3
3 4
4 5
```

Sample Output 2

```
3
1 3 4
```