Data Structure and Algorithm, Spring 2019

Homework 4

Release: Wednesday, June 5, 2019

Due: 23:59:59, Sunday, June 16, 2019

TA E-mail: `dsa1@csie.ntu.edu.tw`

Rules and Instructions

- In homework 4, the problem set contains 4 problems and is divided into two parts, non-programming part (problem 1, 2) and programming part (problem 3, 4).
- Please go to *DSA Judge* (`https://dsa2019.csie.org`) and complete the first problem to familiarize yourself with the usage of the judge system.
- For problems in the non-programming part, you should combine your solutions in ONE PDF file, and then submit it via the judge system. Your file should be as clean as possible, and the use white text on black background is prohibited. Your solution must be as simple as possible. At the TAs' discretion, solutions which are too complicated will be regarded as incorrect. Moreover, if you would like to use any theorem which is not mentioned in the classes, please include its proof in your solution.
- For problems in the programming part, you should write your code in C programming language, and then submit it via the judge system. You can submit up to 15 times per day for each problem. The compilation command of the judge system is `gcc main.c -static -std=c11 -O2 -lm`.
- Discussions with others are encouraged. However, you should write down your solutions in your own words. In addition, for `each problem`, you have to specify the references (the Internet URL you consulted with or the people you discussed with) on the first page of your solution of that problem in your solution.
- The score of the part that is submitted after the deadline will get some penalties according to the following rule:

$$LateScore = (\frac{86400 - DelayTime(sec.)}{86400})OriginalScore$$

Non-Programming Part

Problem 1. k-th Largest Element (28 *points*)

During the lecture, we mentioned that R-B tree can find/add/remove an element in $O(\log n)$-time, where n is the size of the R-B tree. In fact, you can also find the k-th largest element in the R-B tree in $O(\log n)$-time if you maintain more information in the tree.

For each node x, let $sz(x)$ be the number of x's descendants including x itself.

1. (5 *points*) Given a R-B tree of size n, show that you can compute the $sz(x)$ information of all node x in $O(n)$-time.

2. (3 *points*) Given a R-B tree of size n and $sz(x)$ of all node x, show that when you add a new element to the tree and no rotation is performed, you can update the $sz(x)$ information of all node x in $O(\log n)$-time.

3. (3 *points*) Given a R-B(x) tree of size n and $sz(x)$ of all node x, show that when you remove an element in the tree and no rotation is performed, you can update the $sz(x)$ information of all node x in $O(\log n)$-time.

4. (7 *points*) Given a R-B tree of size n and $sz(x)$ of all node x, show that when you do a left-rotate, you can update the $sz(x)$ information of all node x in $O(1)$-time.

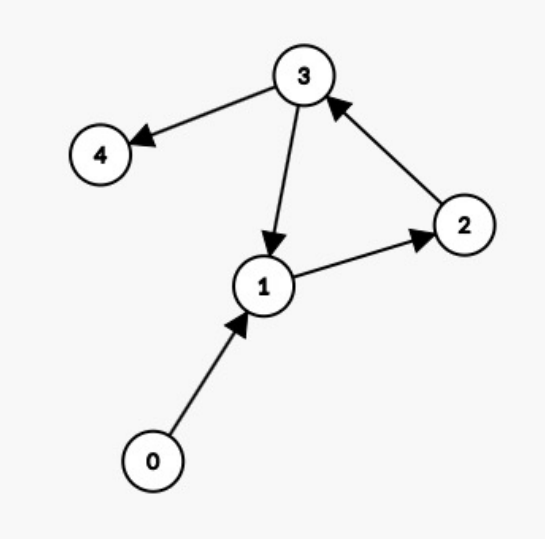
5. (10 *points*) Given a R-B tree of size n and $sz(x)$ of all node x, show that you can find the k-th largest element in the tree in $O(\log n)$-time.

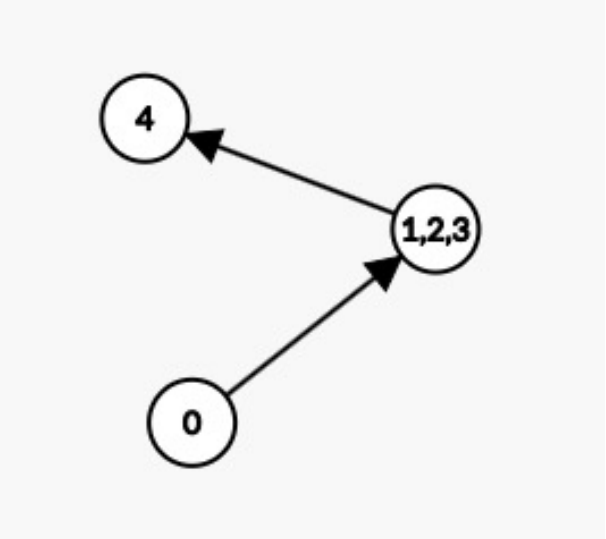
Problem 2. Strongly Connected Component (32 *points*)

Given a directed graph $G = (V, E)$, G is said to be *strongly connected* if every vertex is reachable from every other vertex.In this problem, our goal is to partition the graph into several subgraphs, where all these subgraphs are still strongly connected.

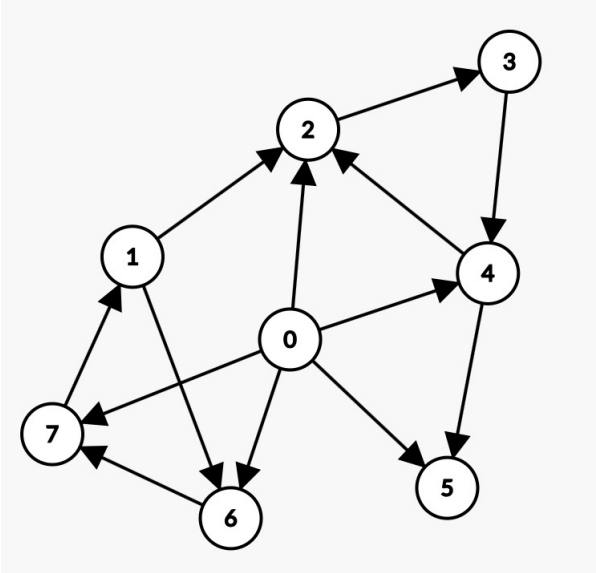
Here, the reverse graph G^R of graph G: G^R is obtained by reversing the directions of all edges in graph G.

In the following questions, we will reduce the size of some graph using the definition of strongly connected component. Here is a simple example.





The vertices 1, 2, 3 can be merged into one vertex since there are strongly connected (every vertex is reachable from every other vertex). Therefore, the left graph above can be reduced to the right graph.



1. (7 *points*) Reduce the graph above with the following rules and draw the result.

   - A subgraph can be reduced to a vertex if it is strongly connected.
   - The graph you draw should be a DAG (Directed Acyclic Graph).

2. (10 *points*) Denote the reduced graph as $\overline{G}$. Prove that $\overline{G^R} = \overline{G}^R$.

3. (15 *points*) Kosaraju's algorithm is a linear-time algorithm to find the strongly connected components of a directed graph. The algorithm can be described with the scenario below,

- Create an empty stack and perform DFS traversal of a graph. In DFS traversal, *after* calling recursive DFS for adjacent vertices of a vertex, push the vertex to stack.
- Pop a vertex from S one by one. Take the popped vertex as source and perform DFS on G^R. The vertices that this DFS visited are the strongly connected components of the source vertex.
- Remove these vertices from G^R (or mark them as visited), and continue popping a vertex from S, performing DFS over again. Finally, obtain all strongly connected components.

Based on the scenario above, write down the C-code of the Kosaraju's algorithm. The goal of the algorithm is to build a mapping from a vertex to the strongly connected component it belongs to. For example, if vertex 1, 2, 3 are in the same components, then $SCC[1], SCC[2], SCC[3]$ should be the same where $SCC[i]$ refers to the components that vertex i belongs to.
Make sure to clearly define all the variables you need. The complexity should be linear, i.e. if there are V vertices and E edges, the complexity is $O(V + E)$

Programming Part

Problem 3. Game of Pancakes (20 *points*)

Your friend Mama is playing a mobile game. He is so intelligent that you cannot beat him. So you're going to write a program to play the game for you.

The game is defined as follows:

There are a lot of pancakes on a bidirectional graph. Each pancake has a value between 1 and $L-1$ (inclusive). At the beginning of the game you're located at vertex 0 and your level is 1. When your level is i, you can only eat pancakes with value i and you can only pass through the vertices that are either empty or have value i. After you leave a vertex, the pancakes on that vertex will regenerate immediately. It takes one step to move between an edge or eat a pancake. Your level upgrades by 1 after you eat a pancake. The goal of the game is to reach level L. Please compute the minimum steps you need to walk to reach level L.

Input

The first line have 4 integers V, E, L, G, which are defined as follows:

- V: The number of vertices.
- E: The number of edges.
- L: The target level of the game.
- G: The number of pancakes on the graph.

Then, the following G lines are two integers p_i, v_i.

- p_i: The index of the vertex pancake i is located at.
- v_i: The value of pancake i.

Then, each of the following E lines has two integers a_i, b_i, which represents an edge between vertex a_i and b_i.

Output

Your program should output an integer, the number of steps you need to walk to reach level L. If you can never reach level L, you should output -1.

Constraint

- $2 \leq V \leq 5 \times 10^4$
- $1 \leq E \leq 10^5$
- $2 \leq L \leq 100$

- $1 \leq G \leq 10^5$
- $0 < p_i < V$ (There's no pancake on vertex 0)
- $1 \leq v_i < L$
- $0 \leq a_i, b_i < V$
- $a_i \neq b_i$
- The graph is connected.

Subtasks

Testgroup0 (40%): $V \leq 20$, $E \leq 20$, $L \leq 10$, $G \leq 20$

Testgroup1 (60%): No other constraints.

Sample Input 1

```
10 10 6 5
1 1
2 2
3 3
4 4
5 5
0 1
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9
9 0
```

Sample Output 1

```
10
```

Sample Input 2

```
10 10 6 5
9 1
```

```
2 2
3 3
4 4
5 5
0 1
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9
9 0
```

Sample Output 2

```
12
```

Sample Input 3

```
10 10 6 9
9 1
8 2
7 3
1 1
2 2
3 3
3 2
4 4
5 5
0 1
1 2
2 3
3 4
4 5
5 6
6 7
7 8
```

```
8 9
9 0
```

Sample Output 3

```
-1
```

Problem 4. Pancakes' Names (20 *points*)

In Arvin's pancake house, there are lots of pancakes. Every pancake has its name. Arvin is wondering the number of pancakes with names between s_1 and s_2. A pancake with name s is between s_1 and s_2 if and only if the lexicographical order of s_1 is larger than or equal to that of s_1 and smaller than or equal to that of s_2.

Your job is to write a program to help Arvin. You're given the list of pancakes. And the list of queries by Arvin. You will need to answer Arvin's question.

Input / Output Format

Since the input may be large, you need to include a file `pancake_names.h`. And call `InitPancakes()` at the beginning.

In `pancake_names.h`, the following data is given:

- `int N`: The number of pancakes
- `char **S`: A string array of size `N`. The names of the given pancakes.
- `int M`: The number of queries.
- `char **Q1, **Q2`: Two string arrays of size `M`. `Q1[i]` and `Q2[i]` means a query.

After you found out the answer for each query. You need to call the function `void AnswerArvin(int i, int count)`. `i` represents the index of the query starting from 0 and `count` represents the number of pancakes whose names are between `Q1[i]` and `Q2[i]`.

After you ansered all the queries, you need to call the function `void ByeArvin()`. And the program will be terminated (the function will not return).

`InitPancakes` runs in less than 3000 microseconds. `AnswerArvin` runs `M` times in less than 100 microseconds. `ByeArvin` runs in less than 100 microseconds. All the data in `pancake_names.h` uses about 467 MiB of memory.

Example

The following code will solving the problem correctly but not efficiently. (It will get TLE but not WA)

```
#include <stdio.h>
#include <string.h>
#include "pancake_names.h"
int main() {
    InitPancakes();
    for (int i = 0; i < M; ++i) {
        int cnt = 0;
```

```
        for (int j = 0; j < N; ++j) {
            if (strcmp(S[j], Q1[i]) >= 0 && strcmp(S[j], Q2[i])
               <= 0)
                ++cnt;
        }
        AnswerArvin(i, cnt);
    }
    ByeArvin();
}
```

You can download an example `pancake_names.h` here.

Constraint

- $2 \leq$ `N` $\leq 2 \times 10^7$
- $2 \leq$ `M` $\leq 2 \times 10^7$
- $1 \leq$`strlen(S[i])`, `strlen(Q1[i])`, `strlen(Q2[i])` ≤ 4
- `Q1[i]` is smaller than or equal to `Q2[i]` lexicographically.
- `S`, `Q1`, `Q2` contain lowercase English alphabets.

Subtasks

Testgroup0 (40%): `N` $\leq 10^5$, `M` $\leq 10^5$, `strlen(S[i])`, `strlen(Q1[i])`, `strlen(Q2[i])` ≤ 2

Testgroup1 (60%): No other constraints.