Homework #4

Due Time: 2016/1/12 (Thu.) 14:20
Contact TAs: `ada01@csie.ntu.edu.tw`

Instructions and Announcements

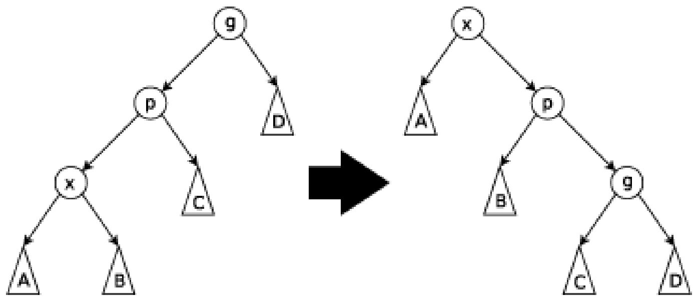
- There are two topic exercises and four regular problems. Topic exercises are easier than regular problems and are part of your mini-homeworks.
- **Programming.** The judge system is located at `https://ada01-judge.csie.org`. Please login and submit your code for the programming problems (i.e., those containing "Programming" in the problem title) by the deadline. NO LATE SUBMISSION IS ALLOWED.
- **Hand-written.** For other problems (also known as the "hand-written problems"), please submit your answers to the instructor at the beginning of the class. NO LATE SUBMISSION IS ALLOWED.
- **Collaboration policy.** Discussions with others are strongly encouraged. However, you should write down your solutions **in your own words**. In addition, for **each and every** problem you have to specify the references (e.g., the Internet URL you consulted with or the people you discussed with) on the first page of your solution to that problem. You may get zero point for problems due to the lack of references.
- Top-graded solutions/codes may be published as references for your fellow classmates.

Problem 1: Splay tree

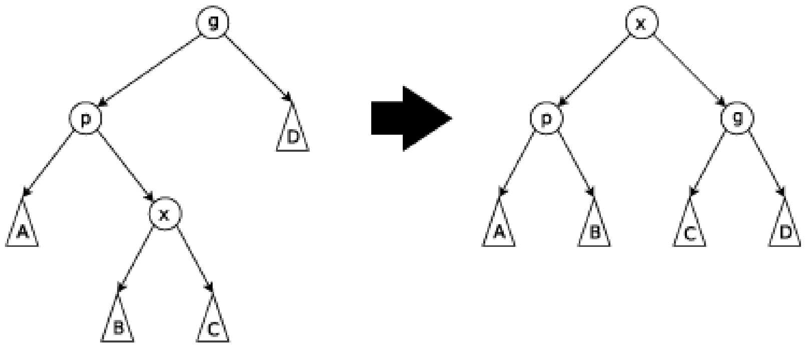
The average search time of a binary search tree is $O(\log n)$. However, in the worst case, it may degrade to $O(n)$. In this problem, we will investigate one kind of self-balancing binary search tree called **splay tree**, which can overcome the above mentioned problem and guarantee $O(\log n)$ amortizd search time in the worst case.

In a splay tree, when a node x is accessed (searched), x will be moved to the root, also called splaying, through a series of *splay steps*. There are three types of splay steps, **zig-zig**, **zig-zag**, and **zig**, each of which re-balances the tree while moving node x upward. A splay step is selected depending on the location of node x, as illustrated below:

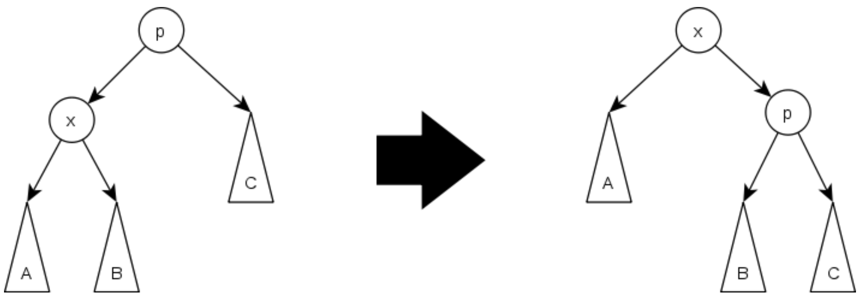
zig-zig step:



zig-zag step:



zig step:



For more details about splay tree, please refer to `https://en.wikipedia.org/wiki/Splay_tree` and `https://www.cs.usfca.edu/~galles/visualization/SplayTree.html`.

A splay tree may become unbalance after certain operations and its worst case search time becomes the same as the naive binary search tree, which is $O(n)$. In spite of the poor worst-case performance, a splay tree performs well in an amortized sense. That is, in a sequence of operations, the average cost of a single operation is $O(\log n)$. The basic idea is that although accessing a deep node is expensive, the subsequent splaying process will make the tree more balanced.

Now you are asked to prove the above argument formally using the **accounting method** of amortized analysis. Firstly, we define following notations:

- size $n(v)$: number of nodes in the subtree rooted at the node v.
- rank $r(v)$: $\log(n(v))$.
- T: a splay tree.
- t: root node.
- $r(T)$: sum of the ranks of all the nodes of T.
- $r(v)$ and $r'(v)$: $r(v)$ is the rank before splaying and $r'(v)$ is the rank after splaying.
- x, p, g: x, p, and g are the target node, its parent, and its grandparent respectively as the figures above.

To use the accounting method, we keep a virtual account at each node to store "credit". Additionally, to make sure there is always enough credit for paying for splaying, we maintain the following invariant:

Each node v of T always has $r(v)$ credit in its account.

Since we only consider the search operation in this problem, you can assume that the accounts is properly set up for searching (through the credit-aware insertions and deletions). Besides, we only consider the cost of splaying since other operations are proportional to it. We define that the actual cost of a zig is 1 and the actual cost of a zig-zig or a zig-zag is 2.

Please solve the following subproblems to show that a $O(\log n)$ amortized cost is sufficient to maintain the invariant and pay for the entire splaying work. Conceptually, an expensive search will make the tree more balanced (less total credit $r(T)$), and the released credit would compensate the cost of the search. On the other hand, a low-cost search will make the tree more unbalanced (more total credit $r(T)$), and the overcharged credit would be deposited in the account of nodes.

(1) (4%) Prove that δ, the variation of $r(T)$ caused by a single zig step, $\leq 3(r'(x) - r(x))$

(2) (4%) Prove that δ, the variation of $r(T)$ caused by a single zig-zig step, $\leq 3(r'(x) - r(x)) - 2$

(3) (4%) Prove that δ, the variation of $r(T)$ caused by a single zig-zag step, $\leq 3(r'(x) - r(x)) - 2$

(4) (4%) Prove that Δ, the total variation of $r(T)$ caused by splaying a node x at depth d, $\leq 3(r(t) - r(x)) - d + 2$

(5) (4%) Explain why the amortized cost of the search operation of the splay tree is $O(\log n)$

Hint for (1): $\delta = r'(x) + r'(p) - r(x) - r(p)$

Hint for (2)(3): $\delta = r'(x) + r'(p) + r'(g) - r(x) - r(p) - r(g)$

Hint for (2)(3): if $a > 0$, $b > 0$, and $c > a + b$ then $\log a + \log b \leq 2 \log c - 2$

Hint for (4): telescoping series

Problem 2

"Ho! Ho! Ho! Merry Christmas!" Christmas is all around and Santa Claus is preparing his list to send out gifts to nice children. However, in your childhood, have you wondered how Santa Claus knows that you are a naughty child or a nice child? The secret is that Santa's Elves are assigned to evaluate every child in towns during the year. To carry out such hard work, Santa Claus quietly sets up Santa Claus Intelligence Agency, also known as SCIA, in towns to manage his Elves. From Santa Claus's point of view, the world can be viewed as a connected graph. Each town is a vertex in the graph, and for every pair of adjacent towns there is an edge between them. To avoid being noticed, Santa Claus sets up as few SCIAs as possible. His rules are as follows. First, there is at most one SCIA in a town. Second, a town that doesn't have a SCIA must be adjacent to at least one town with a SCIA. To thank you for your hard work in this semester and celebrate the holiday, I will kindly tell you that Santa Claus is facing the **dominating-set problem**.

In graph theory, a dominating set for a graph $G = (V, E)$ is a subset V' of V such that every vertex not in V' is adjacent to at least one member of V'. The **dominating-set problem** is to find a minimum dominating set for graph G. In this problem, we only take connected graphs into consideration.

(1) (2%) The related decision problem for the **dominating-set problem** can be defined as $\text{DMT}(G, k)$: $\{$Does graph G have a dominating set of size k?$\}$ Suppose you are given a subroutine $\text{DMT}(G, k)$ which solves the decision problem. Please formulate a polynomial-time algorithm to find the minimum size of dominating set for a connected graph G using this subroutine. You can assume the running time of calling this subroutine is $O(1)$.

(2) (8%) Given a connected graph H, please give a reduction algorithm to determine whether graph H has a vertex cover of size k with subroutine $\text{DMT}(G, k)$ in part(1). The running time of your reduction algorithm should be polynomial in $|V_h|$ and $|E_h|$.

(3) (7%) With part(1) and part(2), prove that the **dominating-set problem** is NP-complete. (Hint: Notice that the **vertex-cover problem** is NP-complete.)

(4) (3%) However, the same problem might be solvable in polynomial time on certain types of graphs such as tree, block graph, etc. Please give a polynomial-time algorithm to solve the **dominating-set problem** in a tree.

Problem 3 - Tippy's visit (Programming)

Time limit : 1s

Description

Tippy is a cute rabbit. He has n friends living in Itomori town. Each of his friends lives in exactly one house in Itomori, numbered from 1 to n. There are one-way (i.e., directional) roads connecting these houses. Note that there could be multiple roads connecting two houses.

One day, Tippy wants to visit his friends in Itomori. He plans to stay at one of his friend's house and visits other friends whose houses are reachable from where he stays. Now the problem is, which house should he stay so that he can visit as many friends as possible while satisfying the following rules?

Tippy moves according to the following rules,

- After his visit, he must come back to the house he stays.
- Tippy has different moving behaviors depending on his mood. When he is happy, he will move *backward*, otherwise he will move *forward*. That is, if there exists a road from house A to house B, then he can move forward from house A to house B when he is in the normal mood, and he can move backward from B to A when he is happy.
- There are some special shops (e.g., the comet souvenir shop) on special roads. When Tippy passes by these shops, his mood will change. That is, if he was happy, then he will become normal. If he was normal, then he will become happy. His mood will not change otherwise.
- Tippy is happy when he starts to visit his friends, and he hopes that he will still be happy when coming back. (If the last road is special, he should have normal mood before entering this road.)

Input Format

The first line contains two integers n, m, indicating the number of houses and roads in the town. In the next m lines, each line contains 3 integers a, b, s, indicating this road is from house a to house b. If $s = 1$, the road is special; otherwise it is a normal road.

- $1 \leq n \leq 10000$
- $1 \leq m \leq 100000$
- $s \in \{0, 1\}$

- Test group 0 (0 points) : Sample testcases.
- Test group 1 (20 points) : $n \leq 650, m \leq 800$.
- Test group 2 (30 points) : $n \leq 3000, m \leq 8000$.
- Test group 3 (50 points) : No additional constraints.

Output Format

Output exactly n lines. In the i-th line exactly one integer, indicating the number of friends he could visit, when he start his visit from house number i.

Sample Input 1

```
5 5
2 1 1
2 3 0
3 4 0
4 2 0
5 3 1
```

Sample Output 1

```
4
4
4
4
1
```

Explanation of Sample Input 1

- If Tippy stays at house number 1, he can visit friends from houses 1, 2, 3, 4.
- If Tippy stays at house number 2, he can visit friends from houses 2, 3, 4, 5.
- If Tippy stays at house number 3, he can visit friends from houses 2, 3, 4, 5.
- If Tippy stays at house number 4, he can visit friends from houses 2, 3, 4, 5.
- If Tippy stays at house number 5, he can visit friends from houses 5.

Problem 4 - The Last and the Easiest Problem

Time limit : 1s

Description

(30%) Since this is the last homework problem, we shall make it an easy one.

So here's the problem. Given an undirected graph $G = (V, E)$, output the longest **simple path**. A **path** is a sequence of vertices $(v_1, v_2, \cdots, v_n)$ such that $(v_i, v_{i+1}) \in E$ for every i. A path is **simple** if it does not have repeated vertices (i.e., $v_i \neq v_j, \forall i \neq j$).

But wait! Isn't this exactly the *Longest path problem*, which is NP-*hard*? This isn't an easy problem at all! So, to make this an easy problem, we would provide you all the inputs here which you could download it. The final test data will be exactly these inputs, but with the label of the vertices shuffled (i.e., the graphs in the testdatas would be isomorphic to these inputs).

Moreover, since this is an NP-hard problem, we don't even know the exact answer either. So if your program outputs a solution with length greater than 80% of my solution, you would receive 60% of the points. And if your program outputs a solution with length greater than 95% of my solution, you would receive full credit.

Input Format

The first line contains two integers V, E, seperated by space, which represent the number of vertices and the number of edges. For the following E lines, each line has two integers u_i, v_i which indicate that there is an edge from u_i to v_i.

- $1 \leq V, E \leq 10^5$
- $1 \leq u_i, v_i \leq V$
- for any $i \neq j$, $(u_i, v_i) \neq (u_j, v_j)$ and $(u_i, v_i) \neq (v_j, u_j)$.

- Test group 0 (0 points) : Sample test cases, and you only needs to output a solution which is better than 80% of our solution.
- Test group 1 – 20 (60 points, 3 points each) : You only needs to output a solution which is better than 80% of our solution.
- Test group 21 – 40 (40 points, 2 points each) : These testdatas are same as test group 1 - 20 (i.e., group #21 = group #1, ...), but you needs to output a solution which is better than 95% of our solution.

Output Format

Output two lines.
In the first line, output a single integer n, which is the length of the path you found. In the second line, output n integer seperated with spaces, which represent the vertices in the path $v_1, v_2, \cdots, v_n$.

Sample Input 1

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

Sample Output 1

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

Sample Input 2

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

Sample Output 2

```
3
3 1 2
```