Homework #1

Due Time: 2014/10/13 (Mon.) 17:00
Contact TAs: `ada@csie.ntu.edu.tw`

Instructions and Announcements

- For the first time submitting your ADA 2014 HW, please create a repository for ADA on `bitbucket` (`https://bitbucket.org`). And share the repository with user `ada2014` with `read` permission.
- You have to login to judgegirl system (`http://ada2014.csie.org`) to bind the bitbucket repository to your account. After you push the source code to repository, you can run the judge and get the score. Please refer to the guide of judgegirl on the index page of judge system for further details.
- For programming problems (those containing **Programming** in the problem topic), commit your source code to your bitbucket repository. You should create a new folder for every question, with names as the problem number on the judgegirl. For example, the source code of problem "`a+b problem`" should be under folder "`1`".
- The filename of the source code should be the problem number on the judgegirl, i.e., the same as the folder name. Once you name your source code properly, you should be able to view your code after logging in to judgegirl. You will get some penalties in your grade if your submission does not follow the naming rule.
  Code comments and documentations are encouraged. To submit a documentation, please upload it to any cloud drive and put the access URL as a comment at the end of the source code.
- For writing problems, submit the answers in an electronic copy via the git repository or in a hard copy to the box placed in CSIE R217 before the due date/time.
- Except the programming assignment, each student may only choose to submit the homework in only one way; either all in a hard copy or all in an electronic copy via git. If you submit your homework partially in one way and partially in the other way, you might only get the score of the part submitted as a hard copy or the part submitted via git (the one that the grading TA chooses).
- If you choose to submit the answers of the writing problems via git, please combine the answers of all writing problems into **only ONE file** in the PDF format, with the file name in the format of "`hw[# of HW].pdf`" (e.g. "`hw1.pdf`"), all in **lowercase**; otherwise, you might only get the score of one of the files (the one that the grading TA chooses).
- 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 to that problem.
- **NO LATE SUBMISSION IS ALLOWED** for the homework submission in hard copies — no score will be given for the part that is submitted after the deadline. For submissions via git (including the programming assignment and electronic copies of the writing problems), up to one day of delay is allowed; however, the score of the part that is submitted after the deadline will get some penalties according to the following rule (the time will be in seconds):

$$\text{LATE\_SCORE} = \text{ORIGINAL\_SCORE} \times (1 - \text{DELAY\_TIME}/86400)$$

Problem 1

(1) Solve the recurrences. Assume that $T(1) = 1$.

(a) (2%) $T(n) = 9T(\lfloor \frac{n}{3} \rfloor) + 10n$
Use the substitution method to prove that $T(n) = O(n^2)$.

(b) (2%) $T(n) = 4T(\frac{n}{2}) + 10n^2$
Use a recursion tree to prove that $T(n) = O(n^2 \log n)$.

(2) Calculate the tight bound (Θ) of the following recurrences. You are allowed to use any methods. Assume that $T(1) = 1$.

(a) (3%) $T(n) = 8T(\frac{n}{2}) + 2n^3 \log n + 2n^2$

(b) (3%) $T(n) = 5T(\frac{n}{2}) + n^3$

(3) (5%) Playing jigsaw puzzles is one of Michael's hobbies. After solving many difficult puzzles, he wants to estimate the time he needs to solve a puzzle of a specific size.

To solve a puzzle with k pieces, Michael first divides them into 4 piles. After solving each of the 4 piles, he then simply puts them together to complete the puzzle. He also applies this strategies recursively to the smaller piles unless there is just 1 piece in the pile.

Usually, Michael can correctly classify the puzzle pieces into piles in k seconds. However, due to color weakness, for small piles with $k < s$ pieces, Michael can hardly tell the colors of different pieces apart so that it instead takes k^2 seconds to complete the classification.

Assume that Michael can always put 4 complete piles of puzzles into one in constant times. Please calculate the tight bounds (Θ) of the time he needs to finish a puzzle, given the number of pieces in the puzzle k and the color weakness threshold s. Please write down the recurrences and derive its solution.

Problem 2

(1) (3%) We have learned how to solve the "Maximum Subarray Sum" problem in the class. Please design a Divide and Conquer algorithm to find "Maximum Subarray Product" in a size-N array within $O(N \log N)$ time. (Notice that elements in the array are real numbers.)

(2) (3%) In the class, we learned how to find the closest pairs of points in 2D space. In fact, the same approach can also be applied into the case of 3D space. In the case of 2D space, we divide points by one line. For 3D space, we divide points by one plane. In the stage of conquer, the case of 2D space claims that there are a constant number of points within a d-by-$2d$ rectangle, where d is minimum distance of pairs of points in the same side. Like the case of 2D space, in the case of 3D space, please prove that there are a constant number of points within a d-by-$2d$-by-$2d$ cuboid.

(3% for bonus) Please design an algorithm to find "Closest Pair of Points in 3D Space" given N points in the 3D space within $O(N \log N)$ time. Brifly argue the correctness of your algorithm and show that it indeed runs in $O(N \log N)$ time.

(3) (4%) If you want to find the closest point for every point, the method we mentioned in the class cannot solve this problem in time complexity $O(N \log N)$. Please explain why it cannot solve this problem in $O(N \log N)$. If you can design another algorithm to solve this problem in time complexity $O(N \log N)$, you will also get full points.

Problem 3

Given a sequence of numbers $\langle a_n \rangle$ of length N, we define *consecutive sum* as the numerical summation of a consecutive sub-sequence. For example, the sequence $\langle a_n \rangle = \langle 1, 2, 5, 14, 42 \rangle$ has a consecutive sum $\sum_{i=2}^{3} a_i = 2 + 5 = 7$ that corresponds to consecutive sub-sequence $\langle 2, 5 \rangle$. Note that a consecutive sub-sequence must be consecutive in the original sequence. For instance, the sequence $\langle 1, 5, 42 \rangle$ is NOT a consecutive sub-sequence of $\langle a_n \rangle$, although it is a simple sub-sequence of $\langle a_n \rangle$.

In this problem, we will learn how to discover the k-th largest consecutive sum of a sequence with a pre-defined k. Take the sequence $\langle a_n \rangle$ in the previous paragraph as an example, the consecutive sum $\sum_{i=2}^{5} a_i = 2 + 5 + 14 + 42 = 63$ is the 2-nd largest consecutive sum.

(1) (2%) Design an efficient algorithm to find the consecutive sum of any consecutive sub-sequence in time complexity $O(1)$ with $O(N)$ pre-processing, given the start index and the end index of the consecutive sub-sequence.

(2) (2%) Design a naïve brute-force algorithm to calculate the k-th largest consecutive sum of a sequence in time complexity $O(N^2 \log N)$. The algorithm should be written in pseudo code or C code. Briefly argue the correctness of your code and show that it indeed runs in $O(N^2 \log N)$.

(3) (3%) An inversion of a number sequence $\langle b_n \rangle$ is defined as a pair of two number $(b_i, b_j); b_i, b_j \in \langle b_n \rangle$ if and only if $i < j$ and $b_i > b_j$. Let $I(\langle b_n \rangle)$ be the set of inversions in $\langle b_n \rangle$. For example, if $\langle b_n \rangle = \langle 1, 3, 4, 2 \rangle$, $I(\langle b_n \rangle) = \{(3, 2), (4, 2)\}$. The number of inversions $|I(\langle b_n \rangle)|$ is 2.

Consider the sequence $\langle b_n \rangle = \langle p_n \rangle \langle q_n \rangle$ of length N. It shows that $\langle b_n \rangle$ is the concatenation of two sequence $\langle p_n \rangle$ and $\langle q_n \rangle$. For instance, if $\langle p_n \rangle = \langle 1 \rangle$ and $\langle q_n \rangle = \langle 3, 4, 2 \rangle$, $\langle b_n \rangle = \langle p_n \rangle \langle q_n \rangle = \langle 1, 3, 4, 2 \rangle$.

Given a sequence $\langle b_n \rangle = \langle p_n \rangle \langle q_n \rangle$, please design a function $F(\langle p_n \rangle, \langle q_n \rangle)$, which runs in $O(N \log N)$ time, such that

$$|I(\langle b_n \rangle)| = |I(\langle p_n \rangle)| + |I(\langle q_n \rangle)| + F(\langle p_n \rangle, \langle q_n \rangle).$$

The function should be written in pseudo code or C code. You have to **prove** that with your function the above equation holds and briefly argue that it indeed can be calculated in $O(N \log N)$.

(4) (3%) Given an **arbitrary unsorted** sequence $\langle b_n \rangle$ with N numbers, design an efficient algorithm to calculate the number of inversions $|I(\langle b_n \rangle)|$ in $O(N \log N)$ time. You have to **prove** that your algorithm indeed runs in $O(N \log N)$.
Hint 1. Divide and Conquer is powerful.
Hint 2. The merge sort algorithm may be a big help.

(5) (4%) Given a sequence $\langle a_n \rangle$ and a number m, design an efficient algorithm to calculate the number of consecutive sub-sequence whose consecutive sums are greater than m (i.e., the number of consecutive sub-sequences $\langle a_i, a_{i+1}, \cdots, a_j \rangle$ such that $\sum_{k=i}^{j} a_k > m$). Your algorithm should run in $O(N \log N)$. The algorithm should be written in pseudo code or C code. Briefly argue the correctness of your code and show that it indeed runs in $O(N \log N)$.
Hint. Hints in the problem 3.4 are still useful.

(6) (6%) Design an efficient algorithm to calculate the k-th largest consecutive sum of a sequence with a predefined k. Your algorithm should run in $O(N \log N \log R)$, which R is the possible range of values of a consecutive sum of the sequence. The algorithm should be written in pseudo code or C code. Briefly argue the correctness of your code and show that it indeed runs in $O(N \log N \log R)$.
Hint. Think about why the time complexity of this problem has one more $O(\log R)$, compared to the algorithm in the problem 3.5.

Problem 4

Skyland is a big kingdom with a wide territory. Since the territory is large, railway undoubtedly becomes a popular transportation service for long-distance travels. Currently, there are n towns in Skyland, and there are exactly $n-1$ railroads connecting these towns. Each railroad runs between two towns directly. The railroads make all the towns in Skyland *connected*. That is, travelers can move between any two towns via the existing railroads.

According to this railway network structure, the government finds out that some of the pairs of town are too far away through the railways. It will be better if the government adds more railroads. Therefore, the government want to investigate and figure out "how many pairs of towns have distance more than k, through the railway network". The length of each railway is given and k is a pre-defined threshold constant.

(1) (2%) Show that the structure of Skyland railway network can be represented as a *weighted tree*. You should explain your modeling. (i.e., What are the nodes, edges, weight of edges of your tree?)

(2) (3%) Suppose that the railway network in Skyland is a *chain*. That is, each town has no more than two neighborhood towns which are directly connected by the railways. Under this structure assumption, please design an algorithm to solve the problem within $O(N \log N)$ time. In addition to the time complexity, please also prove the correctness of your algorithm.

(3) (5%) Prove that there always exists a node v such that its largest subtree has size no greater than $N/2$ when we let v be the root of the tree, for any arbitrary tree with size N. (Here we define the size of a tree as the number of nodes.) Then, design an algorithm that can find this special node on any given tree within $O(N)$ time. In addition to the time complexity, please also prove the correctness of your algorithm.

(4) (5%) Design an algorithm that can solve Skyland government's problem within $O\big(N(\log N)^2\big)$ time. In addition to the time complexity, please also prove the correctness of your algorithm.

Hint 1. Every pair of nodes on the tree has "exactly one" simple path that connects them.
Hint 2. Divide and conquer. However, the time complexity should be handled carefully.
Hint 3. Use the conclusion from Problem 4.4.

Problem 5 - Boxes (Programming - Basic)

Description

(20%) You received n boxes as a new semester gift from your best friends. The size of box i is $w_i \times h_i \times 1$. Since n may be very large, you want to save space by putting one box into another box. Note that you can rotate boxes freely, and box x can be put into box y if and only if there is a rotation such that each side of box x is smaller than or equal to the corresponding side of box y.

For example, you can put a $1 \times 3 \times 1$ box into a $3 \times 2 \times 1$ box, but you cannot put a $5 \times 7 \times 1$ box into a $6 \times 6 \times 1$ box. Now your are curious about how many ordered pairs of boxes (x, y) such that box x can be put into box y.

Input Format

The first line contains an integer T indicating the number of test cases. Each test case starts with a line containing one integer n, specifying the number of boxes. Each of the following n lines contains two integers w_i and h_i specifying the size of box i.

- $1 \leq T \leq 10$
- $1 \leq n \leq 100000$
- $1 \leq w_i, h_i, \leq 10^9$

Output Format

For each test case, please output the number of ordered pairs (x, y) such that box x can be put into box y. Note that this number could be very large, do not forget to use `long long` and `%lld` in `printf`.

Sample Input

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

Sample Output

```
2
5
```

Problem 6 - MineCrash (Programming - Advanced)

Description

(20%) You and your best friends love to play the most popular game **MineCrash**. In the game, there are some gold blocks in a row. The i-th gold block is occupied by a miner belonging to player b_i. Some events (l, r, v) may happen during the game, such that each miner occupying the l-th to the r-th gold block gain v units of gold. The goal of player i is to collect g_i units of gold. As an intelligent programmer, you want to make your own implementation of the game, and find out when will each player reach his/her goal.

Input Format

The first line contains a integer T indicating the number of test cases. Each test case starts with a line containing three integers n, m, q, specifying the number of players, gold blocks, and events. The next line contains n integers $g_1, g_2, \ldots, g_n$, specifying the goal of each player. The next line contains m integers, $b_1, b_2, \ldots, b_m$, specifying the ownership of each block. The next q lines list the events in chronological order, each of which contains three integers l_i, r_i, v_i that associates with the i-th event.

- $1 \leq T \leq 10$
- $1 \leq n, m, q \leq 10^5$
- $1 \leq g_i, v_i \leq 10^9$
- $1 \leq b_i \leq n$
- $1 \leq l_i \leq r_i \leq m$

Output Format

For each test case, please output n integers $t_1, t_2, \ldots, t_n$ in one line. It means that player i achieves the goal g_i after event t_i. We simply let $t_i = -1$ if player i cannot achieve the goal g_i even after the last event. Note that trailing spaces are not allowed; please follow the output format exactly.

Sample Input

```
2
6 6 1
3 1 4 1 5 9
1 2 3 4 5 6
1 6 3
3 5 4
15 10 50
1 2 3 1 2
2 5 3
1 4 2
2 4 5
1 5 3
```

Sample Output

```
1 1 -1 1 -1 -1
4 3 -1
```

Hint

1. You can simplify the problem by changing it from finding out **WHEN** the goal(s) will be achieved to **WHETHER** the goal(s) will be achieved, in $O(n + m + q)$ time.
2. The problem above can be achieved by putting a $+v$ value on block l and a $-v$ value on block r for an event (l, r, v).
3. Try to divide on number of events, and then the solution of Hint 1 will be useful.
4. If the complexity of your algorithm is still very high, then it may indicate that you (unnecessarily) consider too many players in the divided sub-problems.