Data Structure and Algorithm, Spring 2023

Homework 2

Due: 13:00:00, Tuesday, May 2, 2023

TA E-mail: dsa_ta@csie.ntu.edu.tw

Rules and Instructions

- Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.
- In Homework 2, the problem set contains a special Problem 0 (see below) and 5 other problems. The set is divided into two parts, the non-programming part (Problems 0, 1, 2, 3) and the programming part (Problems 4, 5).
- For problems in the non-programming part, you should combine your solutions in *one* PDF file. Your file should generally be legible with a white/light background—using white/light text on a dark/black background is prohibited. Your solution must be as simple as possible. At the TAs' discretion, solutions that are too complicated can be penalized or even regarded as incorrect. If you would like to use any theorem which is not mentioned in the classes, please include its proof in your solution.
- The PDF file for the non-programming part should be submitted to Gradescope as instructed, and you should use Gradescope to tag the pages that correspond to each subproblem to facilitate the TAs' grading. Failure to tag the correct pages of the subproblem can cost you a 20% penalty.
- For the programming part, you should have visited the *DSA Judge* (https://dsa2023.csie.org/) and familiarized yourself with how to submit your code via the judge system in Homework 0.
- For problems in the programming part, you should write your code in C programming language, and then submit the code via the judge system. Each day, you can submit up to 5 times for each problem. To encourage you to start early, we allow 10 times of submissions per day in the first week **(from 2023/04/04 to 2023/04/10)**. The judge system will compile your code with

```
gcc main.c -static -O2 -std=c11
```

- For debugging in the programming part, we strongly suggest you produce your own test-cases with a program. You can also use tools like `gdb` or compile with flag `-fsanitize=address` to help you solve issues like illegal memory usage. For `gdb`, you are strongly encouraged to use compile flag `-g` to preserve symbols after compiling. Note: For students who use IDE (VScode, DevC++...) you can modify the compile command in the settings/preference section. Below are some examples:
  - `gcc main.c -O2 -std=c11 -fsanitize=address # Works on Unix-like OS`
  - `gcc main.c -O2 -std=c11 -g # use it with gdb`
- Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.
- Since everyone needs to write the final solutions alone, there is **absolutely no need to lend your homework solutions and/or source codes to your classmates at any time.** In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.
- The score of the part that is submitted after the deadline will get some penalties according to the following rule:

$$Late\ Score = \max\left(\left(\frac{5 \times 86400 - Delay\ Time(sec.)}{5 \times 86400}\right) \times Original\ Score, 0\right)$$

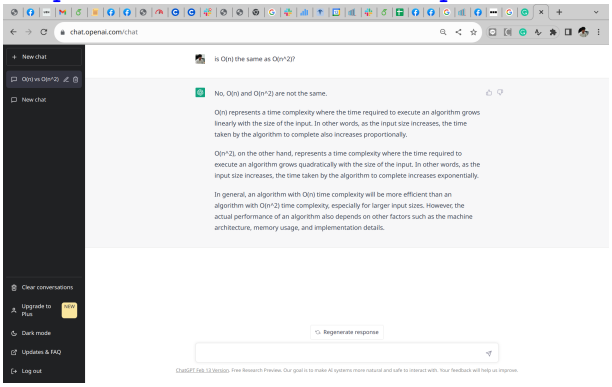
- If you have questions about HW2, please go to the discord channel and discuss (*strongly preferred*, which will provide everyone with a more interactive learning experience). If you really need an email answer, please follow the rules outlined below to get a fast response:
  - The subject should contain two tags, `"[hw2]"` and `"[Px]"`, specifying the problem where you have questions. For example, `"[hw2][P3] Is k in subproblem 3 an integer"`. Adding these tags allows the TAs to track the status of each email and to provide faster responses to you.
  - If you want to provide your code segments to us as part of your question, please upload it to Gist or similar platforms and provide the link. Please remember to protect your uploaded materials with proper permission settings. Screenshots or code segments directly included in the email are discouraged and may not be reviewed.

Non-Programming Part

Problem 0 - Proper References (0 pts)

For each problem below, please specify the references (the Internet URL you consulted with or the classmates/friends you discussed with) in your PDF submitted to Gradescope. If you have used chatGPT or similar tools, please provide a quick snapshot of your interaction with the tools. *You do not need to list the TAs/instructors.* If you finished any problem all by yourself (or just with the help of TAs/instructors), just say "all by myself." While we did not allocate any points to this problem, failure to complete this problem can lead to penalty (i.e. negative points). Examples are

- Problem 1: Alice (B86506002), Bob (B86506054), and https://stackoverflow.com/questions/982388/



- Problem 2: all by myself
- ...

Listing the references in this problem does *not* mean you could copy from them. We cannot stress this enough: *you should always write the final solutions alone and understand them fully.*

Problem 1 - Birds Playing in the Tree (100 pts)

As you walk out of the CSIE building following the DSA class, you come across a pigeon and a spotted dove cooing about the problems discussed during the lecture on the tree data structure. Despite their best efforts, the birds are struggling to solve some of the more complex problems. Intrigued by the challenge, you decide to help those birds by solving these problems yourself.

1. (20 pts) Assume that n_i is the number of nodes with degree i ($0 \leq i \leq k$) in a tree, prove that $n_0 = 1 + \sum_{i=1}^{k} n_i(i-1)$.

2. (10 pts) Given the inorder and preorder traversals of a binary tree as follows.

   ```
   inorder traversal: 4, 26, 19, 22, 7, 15, 34, 11, 13, 8
   preorder traversal: 22, 4, 19, 26, 11, 15, 7, 34, 13, 8
   ```

   The pigeon claims it can reconstruct a unique binary tree by only the inorder and preorder traversals. If you believe that the pigeon's claim is reasonable, describe a "human" (bird) algorithm for doing so and draw the reconstructed tree. Otherwise, explain why the reconstruction to a unique binary tree is not possible. We encourage you to use drawings to help your explanation—a picture is worth a thousand words.

3. (10 pts) The spotted dove, on the other hand, claims it can reconstruct a unique binary tree by only the preorder and postorder traversals.

   ```
   preorder traversal: 22, 4, 19, 26, 11, 15, 7, 34, 13, 8
   postorder traversal: 26, 19, 4, 7, 34, 15, 8, 13, 11, 22
   ```

   If you believe that the spotted dove's claim is reasonable, describe a "human" (bird) algorithm for doing so and draw the reconstructed tree. Otherwise, explain why the reconstruction to a unique binary tree is not possible. We encourage you to use drawings to help your explanation—a picture is worth a thousand words.

4. (10 pts) Given a sorted array A that contains $[1, 7, 8, 12, 26, 34, 52, 76, 84]$, construct a shortest Binary Search Tree (BST) from those numbers and draw the tree. No other explanations are needed.

5. (20 pts) Given a sorted array A of size n in the ascending order, design an $O(n)$ algorithm that constructs a shortest BST. Briefly explain why your algorithm works correctly within the designated time complexity. You can choose to either use pseudocode with no more than 30 lines, or use English/Chinese to describe the core design of your algorithm. You do not need to rigorously prove the correctness nor the time complexity (yeah!!).

After the discussion, the pigeon and the spotted dove find N different alphabet bugs in front of them. Each bug can be identified by a string of P lowercase alphabetic characters. Both birds have built-in memory on their top M favorite bugs, stored in some data structure. Then, upon seeing one of the N bugs, the bird needs to search for whether the bug matches any of the M bugs in their memory. Note that there can be redundancy in their memories.

6. (10 pts) Analyze the time and extra-space complexity of Pigeon's Algorithms, for storing and searching.

7. (10 pts) Analyze the time and extra-space complexity of Spotted Dove's Algorithms, for storing and searching.

8. (10 pts) Come up with a scenario that you would choose Pigeon's Algorithms over Spotted Dove's, and defend your choice as if you are trying to convince your boss. This is an open question. Use your creativity!

Pigeon's Algorithms

PIGEON-STORE(M, P)

```
1  allocate an M by P array called bugs
2  for i = 1 to M
3      READ a bug into a length-P array s // O(P) time
4      for j = 1 to P
5          bugs[i][j] = s[j]
6  return bugs
```

PIGEON-SEARCH($N, M, P,$ an M by P array $bugs$)

```
 1  allocate a size-N array called eat and initialize every element to FALSE
 2  for i = 1 to N
 3      READ a bug into a length-P array s // O(P) time
 4      for j = 1 to M
 5          match = TRUE
 6          for k = 1 to P
 7              if bugs[j][k] is not equal to s[k]
 8                  match = FALSE
 9                  continue to Line 2
10          if match is equal to TRUE
11              eat[i] = TRUE
12              continue to Line 2
13  return eat
```

Spotted Dove's Algorithms

SPOTTEDDOVE-STORE(M, P)

```
1  declare a structure called Node, which can be linked to at most 26 other nodes.
   // The links are stored in child using 'a' to 'z' as its index.
2  allocate a Node called bugsRoot with every child element set to NIL // O(1) time
3  for i = 1 to M
4      READ a bug into a length-P array s // O(P) time
5      let a cursor cur link to bugsRoot
6      for j = 1 to P
7          idx = s[j]
8          if cur.child[idx] is equal to NIL
9              allocate a new Node new with every child element set to NIL // O(1) time
10             cur.child[idx] = new
11         cur = cur.child[idx]
12 return bugsRoot
```

SPOTTEDDOVE-SEARCH($N, M, P,$ the *bugsRoot* tree stored above)

```
1  allocate a size-N array called eat and initialize every element to FALSE
2  for i = 1 to N
3      READ a bug into a length-P array s // O(P) time
4      match = TRUE
5      let a cursor cur link to bugsRoot
6      for j = 1 to P
7          idx = s[j]
8          if cur.child[idx] is equal to NIL
9              match = FALSE
10             continue to Line 2
11         cur = cur.child[idx]
12     if match is equal to TRUE
13         eat[i] = TRUE
14         continue to Line 2
15 return eat
```

Problem 2 - Valorant (100 pts)

Valorant is an enjoyable First-Person Shooter (FPS) game with many players, each of which is assigned a rank level ranging from 0 (iron) to 9 (radiant). Each round of the game divides 10 players into two teams, the Attacker and the Defender. After completing your DSA programming homework, you really want to have some fun with Valorant!

As you prefer not to be paired up with random strangers, you start searching through your friend list in order to find some reliable teammates for your Valorant game. However, your extensive list of friends is simply too long to scan through manually. Consequently, you decide to sort your friend list according to their respective rank levels.

1. (10 pts) Assume that you have eight friends with ranks $[7, 3, 5, 0, 2, 8, 6, 1]$ stored in an array. Run the bottom-up merge sort algorithm on the array and illustrate each step of the algorithm.

After selecting your teammates, you decide to review the rank history of one of your teammates, Sova, while waiting to start the game.

2. (10pts) Sova's rank history for each season is stored in an array $r = [6, 5, 9, 2, 0, 5, 8, 1]$, where 6 is zir rank in the first season, 5 is zir rank in the second season, and so on. Sometimes the rank increases in future seasons, like from Season 2 to Season 3. Sometimes it decreases, like from Season 1 to Season 5. For two seasons i and j with $i < j$, a reversion in rank means that $r[i] > r[j]$. Please count the total number of reversions. Briefly illustrate how you do the calculation.

For the three sub-problems listed below, please ensure that your answers consist of the following components:

(a) Please describe your algorithm using pseudocode and provide a **brief explanation** of why it works. Your pseudocode for each sub-problem should be concise and easy to read, **consisting of no more than 30 lines**. Please note that overly complex or lengthy pseudocode may result in point deductions.

(b) Analyze the time complexity and *extra-space* complexity of your algorithm. Please note that if your algorithm is not efficient in terms of time or space, you may not receive full credit for your solution.

Because Sova's rank history contains too many reversions, you suspect that ze might be illegally trolling or cheating. You thus decide to report Sova to increase the overall fairness of the game (just like what you should do for our class!), and start inspecting the rank history of your other teammates.

3. (20 pts) Given an array r of size n that stores the rank history for some player from season 1 to season n. Design an $O(n \log n)$ time and $O(n)$ extra-space algorithm that counts the number of reversions within r.

4. (25 pts) After having fun in your game, you discovered that your friend Yoru appears to be upset, as ze has been struggling with poor game performance lately. You thus decide to help zir by searching for a more skilled player, such as Jett, to pair with zir. Actually, you aim for a higher goal—for each player in the game, you hope to identify some more skilled players to pair with zir.

   In addition to the rank r, each player carries another value called the Kill/Death (K/D) ratio. A more skilled player is defined as someone with a better rank **and** a better K/D ratio. Each player seeks to find some more skilled player as zir pairing candidate. Given the ranks and the K/D ratios of n players, design an algorithm that calculates the total number of possible pairing candidates, summed across all n players. Your algorithm should run with a time complexity of $O(n \log n)$ and an extra-space complexity of $O(n)$.

   For instance, consider five players of the following rank and ratio:

   $$\{[2, 1.4], [3, 1.6], [1, 0.3], [6, 2.3], [4, 1.0]\}.$$

   The first player can be paired two possible candidates, the second or the fourth one; the second player can only be paired with the fourth one. Continuing the calculation, we have $2 + 1 + 4 + 0 + 1 = 8$ possible pairing candidates in total.

   Hint: What you have done in the previous three sub-problems may help!

5. (20 pts) Yoru and Jett decide to analyze their match logs together. In particular, each of their match logs is a sorted array of length n that contains the match scores, ordered from the smallest value to the largest value. Design an algorithm that runs in $O(\log n)$ time and $O(1)$ extra space to help them identify the k-th smallest match scores within the two sorted arrays.

6. (15 pts) Jett find that the code used for sorting the matching logs seems to be running too slowly. The pseudocode is originally written by Jett's friend Neon, which is shown below. Analyze the code and argue why the code does not run with an average time complexity of $O(n \log n)$ on an array of size n.

```
NeonSort(an array named match, oldest, latest)
1  if oldest < latest
2        pi = Partition(match, oldest, latest)
3        NeonSort(match, oldest, pi–1)
4        NeonSort(match, pi + 1, latest)

Partition(an array named match, oldest, latest)
 1  mvp = −1
 2  for j = oldest to latest − 1
 3        if match[j] > mvp
 4              mvp = match[j]
 5  i = oldest–1
 6  for j = oldest to latest − 1
 7        if match[j] < mvp
 8              i = i + 1
 9              Swap(match[i], match[j])
10  Swap(match[i + 1], match[latest])
11  return i + 1
```

Problem 3 - Structure and Algorithm Online (100 pts)

In 2030, Professors Lin and Tsai, together with their NASA team, developed a virtual reality game named Structures and Algorithms Online (SAO). SAO allows students to learn DSA (Data Structures and Algorithms) by immersively playing an RPG game. By wearing a special equipment known as NerveGear, provided by the CSIE student union, and saying "link start!", students can enter the SAO world and enjoy on their learning journey.

However, several weeks after the game's release, some strange incidents occur. Students who are immersed in the game suddenly find that the logout buttons have disappeared. In order to return to reality and enjoy their ordered pizza and soft drinks, they need to conquer the most challenging dungeon in SAO, Aincrad, to retrieve their logout button. As they venture through Aincrad, DSA questions can appear along the way, and they must answer them correctly to progress further in the game. Failure to do so may result in being flunked.

`This might be only a game, but it is not something you play.`

Following some discussions, the stranded students come to a decision to form multiple guilds to minimize the risks involved. Kirito, a member of the Knights of Blood guild, is now prepared to enter Aincrad. However, before he can commence his adventure, he must overcome some hurdles, starting with the seals on the giant doors of the floating city. Kirito must answer the questions posed by these seals to gain access and embark on his journey.

1. (10 pts) Consider an array to be $[4, 8, 7, 6, 3, 74, 10, 16, 11, 6]$, and build a `max heap` by the Build-Max-Heap algorithm listed on page 167 of the textbook (if you do not have a textbook, we will provide the pseudocode on NTU COOL later). Draw the resulting heap. No more explanations are needed—we only need the resulting heap.

2. (10 pts) Consider an array to be $[4, 8, 7, 6, 3, 74, 10, 16, 11, 6]$, and build a `max heap` by inserting the elements into the heap one by one, from the first element of the array to the last one. Draw the resulting heap. No more explanations are needed—we only need the resulting heap.

In RPG games, nothing is more thrilling than discovering a hidden treasure chest. And that is exactly what Kirito has stumbled upon now. "Wow! A hidden treasure chest!" he exclaimed. However, the chest is located on the other side of a canyon. To cross the canyon, there are 15 slates with numbers 1 to 15 engraved on them and a rectangular groove on Kirito's side. The slates can be arranged in a row to perfectly fit into the rectangular groove, forming an array that represents a complete binary tree. If the arrangement satisfies certain conditions, a bridge will appear, connecting the two sides of the canyon and allowing Kirito to reach the chest. Help Kirito pass the canyon by solving the following questions.

3. (20 pts) Assume that 15 different integers are stored in an array. That is, there are 15! possible arrangements. How many of those arrangements can be viewed as a `min heap` when the array represents a complete binary tree? Briefly explain how you reached your answer.

4. (20 pts) Assume that 15 different integers are stored in an array. Among those arrangements that can be viewed as a `min heap`, what is the maximum number of inverse pairs within the array? An inverse pair is defined as indices (i, j) such that $i < j$ but $a[i] > a[j]$. Briefly explain how you reached your answer.

   For example, if the length of the array is 5, then $[1, 2, 5, 4, 3]$ can be regarded as a `min heap`, and $(3, 4)$ is an inverse pair as $a[3] = 5 > a[4] = 4$. The total number of inverse pairs in this array is 3, which is actually the maximum number.

After going through the challenges, Kirito is ready to face the final battle. Before the battle, Kirito wants to upgrade his sword, Elucidator, with some materials that he has on hand. The materials will need to be fused to one big material before upgrading the sword. In each step, Kirito can choose to fuse two materials together, which would cost zir the sum of the two materials' levels while resulting in a new material with a level equal to the very same sum.

5. (20 pts) Assume that Kirito has n materials on hand, with their levels stored in an array *level*, design an algorithm that outputs the minimum cost of fusing all materials together. Your algorithm should run in $O(n \log n)$ time and use $O(1)$ extra space. Explain the algorithm (with pseudocode, or even more preferably, with understandable words) and why it meets the required time and space complexity.

In the final challenge, every guild is prepared to fight with the final boss, the Gleameyes. In particular, each guild has its members sorted by their strength in a *decending* order. Kirito has found a secret spell that can strengthen their defense power, but the spell requires all members from all guilds to be sorted in a *decending* order first.

6. (20 pts) Assume that m guilds, each with n members represented by an array, have been sorted in descending order by their strength. Design an algorithm that combines those m ordered arrays into a big ordered array of length mn, in descending order, with $O(mn \lg m)$ time and $O(m)$ extra space. Explain the algorithm (with pseudocode, or even more preferably, with understandable words) and why it meets the required time and space complexity.

After the guilds conquered all challenges. Now everyone can return to the real world and do their DSA HW2 (and more)! `Congratulations!!`

Problem 4 - Everyone Loves Sweetmelons (100 pts)

Problem Description

Devine Sweetmelon Agency (a.k.a. DSA) is an internationally renowned group company selling sweet melons. Based on credible sources, DSA comprises N companies, numbered from 1 to N, where the company numbered 1 is the topmost company in DSA. The organizational structure of DSA is quite straightforward: the i-th company, except the topmost company, has exactly one parent company x_i.

Chicken-Soup has a strong liking for sweet melons and desires to eat as many sweet melons as possible. To achieve this goal, Chicken-Soup conducts an investigation and obtains the sales plans of DSA for the next M days. According to the sales plans, every day, each company in DSA will launch a sales event that lasts for several days. Specifically, the sales event of company i on day j would offer sweet melons at a price of c_{ij} dollars and will last for d_{ij} days. This implies that Chicken-Soup can purchase sweet melons at the price of c_{ij} on day j, $j+1$, $\ldots$, $j+d_{ij}$.

In order to simplify the purchase process, Chicken-Soup decides to sign a contract with exactly one company in DSA every day. The contract requires ze to buy exactly one sweet melon from the contracting company and each of its offspring companies on the same day. However, despite wanting to purchase all the sweet melons, Chicken-Soup's daily pocket money is limited to C dollars. In addition, Chicken-Soup cannot carry over the daily pocket money to the next day. Given the constraint, can you help calculate the maximum number of sweet melons that Chicken-Soup can purchase each day?

Input

The first line contains three space-separated integers N, M, and C, which are the number of companies, the length of sales events, and Chiken-Soup's daily pocket money, respectively. The second line contains $N-1$ space-separated integers $x_2, x_3, \ldots, x_N$, where x_i represents the parent company of company i. The next input lines can be divided into M parts, each representing the sales plan on the j-th day for $j = 1, 2, \ldots, M$. Each part contains N lines, and the i-th line of the j-th part contains two space-separated integers c_{ij}, d_{ij}, which are the price and the length of the sales plan of company i on day j.

Output

The output should consist of M lines, and the j-th line should be the maximum number of sweet melons that Chicken-Soup can purchase on day j.

Constraints

- $1 \leq NM \leq 10^6$
- $1 \leq C \leq 10^9$
- $1 \leq x_i \leq N$
- $0 \leq d_{ij} \leq M$
- $0 \leq c_{ij} \leq C$

Subtask 1 (10 pts)

- $1 \leq NM \leq 10^3$

Subtask 2 (15 pts)

- $x_i = i - 1, i \in \{2, 3, \ldots, N\}$

Subtask 3 (15 pts)

- $d_{ij} = M$

Subtask 4 (60 pts)

- No other constraints.

Sample Testcases

Sample Input 1

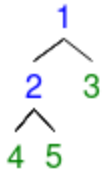
```
5 3 10
1 1 2 2
10 2
1 1
2 2
1 2
8 2
1 2
10 2
1 0
2 1
1 0
5 2
5 1
9 2
3 0
10 1
```

Sample Output 1

```
3
5
1
```

Hints

Here are the details of sample 1. The organization graph of DSA is:



The following table shows the price of sweet melons sold by each company every day.

company	day 1	day 2	day 3
1	10	10, 1	10, 1, 5
2	1	1, 10	10, 5
3	2	2, 1	2, 9
4	1	1, 2	1, 2, 3
5	8	8, 1	8, 10

- On the first day, Chicken-Soup can sign a contract with company 2 and buy sweet melons at prices $1, 1, 8$ from companies $2, 4, 5$ respectively.
- On the second day, Chicken-Soup can sign a contract with company 1 and buy sweet melons at prices $1, 1, 2, 2, 1$ from companies $1, 2, 3, 4, 5$ respectively.
- On the third day, Chicken-Soup can sign a contract with company 5 and buy one sweet melon at price 8 from company 5.

Sample Input 2

```
4 4 20
1 2 3
5 4
10 4
15 4
20 4
20 4
5 4
10 4
15 4
15 4
20 4
5 4
10 4
10 4
15 4
20 4
5 4
```

Sample Output 2

```
1
1
3
4
```

Problem 5 - Magical Cats (100 pts)

Problem Description

Demeter Sphynx Abyssinian (a.k.a. DSA) is a passionate cat lover who shares zir home with hundreds of thousands of furry felines. These adorable creatures consume tons of food every day, so DSA wants to gather statistics in order to formulate a feeding plan for the future.

There are N cats in the house, and each cat living in the i^{th} room has a *unique* appetite a_i. This appetite value represents the amount of food distributed to that cat every day. Furthermore, each cat is associated with a color represented by an integer value b_i.

DSA is interested in establishing a correlation between a cat's color and its appetite. In particular, ze wants to determine the number of cats that have an appetite falling within the range $[l_j, r_j]$ and possess the color c_j.

Nevertheless, there are some greedy cats in the household who desire more food and sneakily exchange their bowls with some unsuspecting cats. To avoid getting caught by DSA, a greedy cat always picks a victim whose appetite is closest to its own from among the cats with larger appetites. Whenever two cats' bowls are swapped, their appetites are swapped as well, since DSA distributes food based on the size of the bowls. Please note that the cat that has the largest appetite at any point cannot be greedy.

Recently, a magical fairy pays a visit to DSA's home and casts some spells on the cats. The fairy first selects a particular color c_j and then picks the cat with the highest or lowest appetite among all the cats with that color. The chosen cat will then be able to increase its appetite to the highest appetite of all cats plus one, denoted as $\max_{0 \le i < N}(a_i) + 1$. However, on occasion, the magic may fail, causing the opposite effect that decreases the cat's appetite to the smallest appetite of all cats minus one, denoted as $\min_{0 \le i < N}(a_i) - 1$. The selection of the cat with the highest (1) or lowest (0) appetite to receive the magic is represented by $s_j \in \{1, 0\}$, and the success (1) or failure (0) of the magic is represented by $t_j \in \{1, 0\}$. It is important to note that a cat's appetite may become negative as a result of this magic.

Given M mixed steps of DSA's question-asking, greedy cats' swapping, and fairy's magic spells, can you make sure that each of DSA's questions is answered correctly?

Input

The first line contains two space-separated integers N, M, the number of cats and the number of steps. The cats will be 0-indexed. The second line contains N space-separated integers $a_0, a_1, \ldots, a_{N-1}$, where a_i is the appetite of the i^{th} cat. The third line contains N space-separated integers $b_0, b_1, \ldots, b_{N-1}$, where b_i is the color of the i^{th} cat. Each of the next M lines contains one of the following:

- 1 c_j l_j r_j: a questioning step (1) followed by three space-separated integers c_j l_j r_j, which denote the color, the lower bound, and the upper bound of the question. There can be no cats that are of color c_j in a questioning step.
- 2 k_j: a swapping step (2) followed by an integer k_j, which is the index of greedy cat.
- 3 c_j s_j t_j: a magic step (3) followed by three space-separated integers c_j s_j t_j, which denotes the color, the direction of the magic, and the success/failure of the magic. We guarantee that a cat with color c_j always exists for a magic step.

Output

For each questioning step, output the answer to the question in a new line.

Constraints

- $2 \leq N \leq 3 \times 10^5$
- $1 \leq M \leq 3 \times 10^5$
- $1 \leq a_i, b_i, c_j \leq 10^9$
- $0 \leq k_j < N$
- $s_j, t_j \in \{0, 1\}$
- $-2 \times 10^9 \leq l_j \leq r_j \leq 2 \times 10^9$

Subtask 1 (10 pts)

- $1 \leq M \leq 10^3$

Subtask 2 (10 pts)

- Only questioning steps.

Subtask 3 (30 pts)

- Only questioning and swapping steps. *Hint: You are highly encouraged to conquer this subtask (or even easier ones) first before solving the problem in general.*

Subtask 4 (50 pts)

- No other constraints.

Sample Testcases

Sample Input 1

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

Sample Output 1

```
2
0
1
2
```

Sample Input 2

```
10 7
6 3 2 4 10 5 9 8 7 1
5 8 1 6 7 7 4 9 1 9
2 8
3 1 0 1
1 1 9 11
1 1 8 10
1 1 5 11
2 8
1 1 5 9
```

Sample Output 2

```
1
1
2
1
```