Competitive Coding @ Berkeley

DeCal Lecture #10 - Segment Trees

Announcements?

- Codeforces is down
  - We can't make the new problemset yet
- The previous problemset is extended to Thursday 4/13 by 11:59PM

Motivation

- Given an array ***a*** of length ***n*** we want a data structure than can…
- ***a*** = [5, 8, 10, 14, 8, 2, 1]
- Be **constructed** in no more than O(***n***) cost

Motivation

- Given an array ***a*** of length ***n*** we want a data structure than can…
- ***a*** = [5, 8, 10, 14, 8, 2, 1]
- Be **constructed** in no more than O(***n***) cost
- **Query:** find the maximum (or min/sum) value of any interval in O(log ***n***)

Motivation

- Given an array ***a*** of length ***n*** we want a data structure than can…
- ***a*** = [5, 8, 10, 14, 8, 2, 1]
- Be **constructed** in no more than O(***n***) cost
- **Query:** find the maximum (or min/sum) value of any interval in O(log ***n***)
- **Update:** change the value of any element in ***a*** in O(log ***n***)
    - This task is challenging. For example, prefix sum arrays do not support this

Structure of a segment tree

- Each node of our tree will store the answer to a specific interval of our array ***a*** (of size ***n***)
- A node represents the answer to some interval [***L***, ***R***]
    - Let ***M*** be the midpoint of this interal, ***M*** = (***L*** + ***R***) / 2

Structure of a segment tree

- Each node of our tree will store the answer to a specific interval of our array ***a*** (of size ***n***)
- A node represents the answer to some interval [***L***, ***R***]
    - Let ***M*** be the midpoint of this interal, ***M*** = (***L*** + ***R***) / 2
- The **root node** stores the answer of the **entire array**, the interval [1, ***n***]

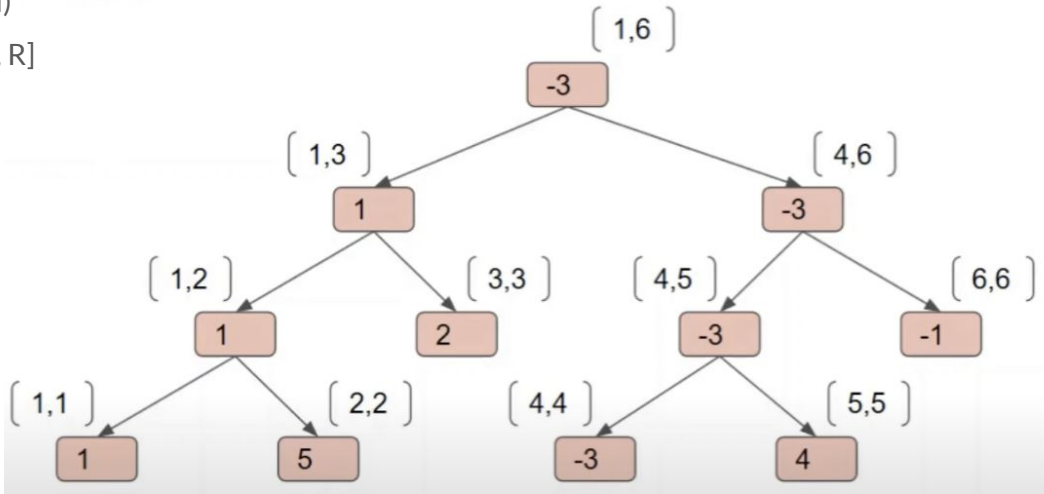
Structure of a segment tree

- Each node of our tree will store the answer to a specific interval of our array ***a*** (of size ***n***)
- A node represents the answer to some interval [***L***, ***R***]
  - Let ***M*** be the midpoint of this interal, ***M*** = (***L*** + ***R***) / 2
- The **root node** stores the answer of the **entire array**, the interval [1, ***n***]
- We construct the other nodes using the following property:
  - Every node with an interval length > 1 has two child nodes:
    - Left child: stores the answer from [***L***, M)
    - Right child: stores the answer from [M, R]

Structure of a segment tree

- Each node represents the answer to some interval [***L***, ***R***]
- Every node with an interval length > 1 has two child nodes:
  - Left child: stores the answer from [***L***, M)
  - Right child: stores the answer from [M, R]

1	5	2	-3	4	-1



Querying

- Let's say we want to find the answer on a segment [***L***, ***R***]
- There are 3 possible cases of any segment node in the tree
    1. The segment has no overlap with [***L***, ***R***]. We should ignore this segment

Querying

- Let's say we want to find the answer on a segment [***L***, ***R***]
- There are 3 possible cases of any segment node in the tree
    1. The segment has no overlap with [***L***, ***R***]. We should ignore this segment
    2. The segment is **completely inside** [***L***, ***R***]. We return the value stored at this node

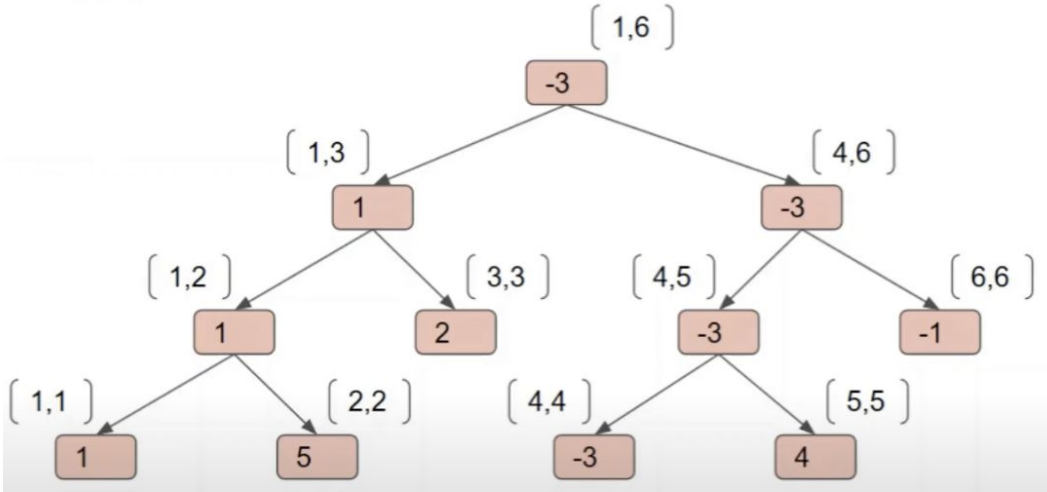
Querying

- Let's say we want to find the answer on a segment [***L***, ***R***]
- There are 3 possible cases of any segment node in the tree
    1. The segment has no overlap with [***L***, ***R***]. We should ignore this segment
    2. The segment is **completely inside** [***L***, ***R***]. We return the value stored at this node
    3. The segment shares **partial overlap** with [***L***, ***R***]. In this case, we make two more recursive calls to the left and right children to find which segments do fit within [***L***, ***R***]
- Works in O(log n) as mentioned earlier

Query Example

- Let's find the minimum value on the segment [3, 5].
  - Assume the array is 1 indexed

1	5	2	-3	4	-1



Update

- We want to update an index ***i*** in to have a different value in our array ***a***
- Observe that ***i*** is only contained in at most **one node** per level of our segment tree

Update

- We want to update an index ***i*** in to have a different value in our array ***a***
- Observe that ***i*** is only contained in at most **one node** per level of our segment tree
- Since the height of the tree is O(log ***n***), this is the most nodes we will have to visit and thus the cost of updating an array element

How to update?

- Start from the root node
- Check if index ***i*** is contained in the left or right child node and recurse to the appropriate one
    - Repeat until we reach our base case of the segment [***i***, ***i***]
    - At this base case, set the node's value to the desired array value

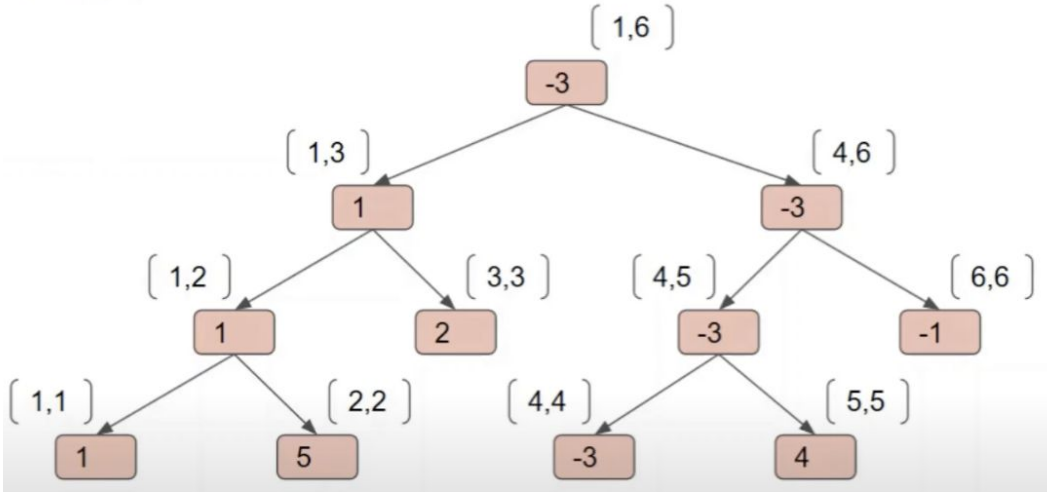
How to update?

- Start from the root node
- Check if index ***i*** is contained in the left or right child node and recurse to the appropriate one
  - Repeat until we reach our base case of the segment [***i***, ***i***]
  - At this base case, set the node's value to the desired array value
- Now when we recurse back, the parent nodes could have been affected by this change!
  - Update parent node's value to be min (or maximum or whatever) of the two child nodes.
- Essentially, we handle out update at the base case, and then work our way up the longer segments that contain ***i***

Update Example

- Let's update index 2 to be equal to a value of -5
  - Assume the array is 1 indexed

1	5	2	-3	4	-1



How to build segment tree?

- Easy method
  - Start with a segment tree where all nodes are empty
  - For all indices ***i*** from [1, ***n***], call the query function to set index ***i*** to a_i

How to build segment tree?

- Easy method
  - Start with a segment tree where all nodes are empty
  - For all indices ***i*** from [1, ***n***], call the query function to set index ***i*** to a_i
  - This works with a cost of O(n log n)
    - Feels extremely wasteful because we have to start from the root of the tree for each update ***n*** times

Linear Time Segment Tree Construction

- We can construct the entire tree from the bottom up similar to querying
- For each node, make two recursive calls to the left and right child nodes
- Finally, we will hit a base case (“segment” of length 1)

Linear Time Segment Tree Construction

- We can construct the entire tree from the bottom up similar to querying
- For each node, make two recursive calls to the left and right child nodes
- Finally, we will hit a base case (“segment” of length 1)
    - Assign the answer to this segment as the value in the array
- After setting this base case, we will recurse back to the parent node
    - We can appropriately set this node’s value as the min/max of its two children because they will both be set
- This time, our construction takes O(n) time

Segment Tree Extra Notes

- You can build a segment tree on any associative function
  - Not just min/max/sum, but also bitwise functions such as XOR or GCD

Segment Tree Extra Notes

- You can build a segment tree on any associative function
  - Not just min/max/sum, but also bitwise functions such as XOR or GCD
- Segment tree is not actually a tree????????

Segment Tree Extra Notes

- You can build a segment tree on any associative function
  - Not just min/max/sum, but also bitwise functions such as XOR or GCD
- Segment tree is not actually a tree????????
  - Surprise! We can store this entire tree inside an array
  - Similar to binary heaps in CS 61B

Longest Increasing Subsequence Revisited

- https://leetcode.com/problems/longest-increasing-subsequence/description/

300. Longest Increasing Subsequence

Medium 16.9K 314

Companies

Given an integer array `nums`, return *the length of the longest* ***strictly increasing*** ***subsequence***.

Example 1:

```
Input: nums = [10,9,2,5,3,7,101,18]
Output: 4
Explanation: The longest increasing subsequence is [2,3,7,101], therefore the length is 4.
```

Example 2:

```
Input: nums = [0,1,0,3,2,3]
Output: 4
```

Example 3:

```
Input: nums = [7,7,7,7,7,7,7]
Output: 1
```

Constraints:

- `1 <= nums.length <= 2500`
- `-10^4 <= nums[i] <= 10^4`

Solution

- We don’t actually build a segment tree on the given array
- Let a node in our segment tree representing [***L***, ***R***] store the maximum length of a LIS ending in a value of ***i*** for all i, ***L*** **<=** i <= ***R***

```cpp
struct segment_tree{
    int size;
    vector<int> tree;

    segment_tree(int n) :size(n){
        tree = vector<int>(4*size);
    }

    void build(vector<int> &v, int i, int l, int r){
        if(r - l == 1){
            tree[i] = v[l];
            return;
        }
        int m = (l+r)/2;
        build(v, 2*i+1, l, m);
        build(v, 2*i+2, m, r);
        tree[i] = max(tree[2*i+1], tree[2*i+2]);
    }

    int query(int i, int l, int r, int lx, int rx){
        if(lx >= r || rx <= l)
            return 0;
        if(l <= lx && rx <= r){
            return tree[i];
        }
        int m = (lx+rx)/2;
        int left = query(2*i+1, l, r, lx, m);
        int right = query(2*i+2, l, r, m, rx);
        return max(left, right);
    }

    void update(int i, int index, int v, int l, int r){
        if(r - l == 1){
            tree[i] = v;
            return;
        }
        int m = (l+r)/2;
        if(index < m){
            update(2*i+1, index, v, l, m);
        }
        else{
            update(2*i+2, index, v, m, r);
        }
        tree[i] = max(tree[2*i+1], tree[2*i+2]);
    }
};
```

C. Chaotic Construction

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

The city Gircle has only one street, and that street is cyclic. This was very convenient in times when people didn't carry a device with compass, GPS and detailed maps around in their pockets, because you only have to walk in one direction and will certainly arrive at your destination. Since Gircle's founding a lot of time has passed. Civil engineers now know a lot more about road network design and most people have immediate access to reliable and accurate navigation systems. However, the passage of time also affected the old street surface and more and more cracks and potholes appeared.

The local government has finally decided to improve the situation, but preserving the city's historic appeal and building new streets are unfortunately mutually exclusive. Because tourism is vital for Gircle's economy, the government's only viable option for improving the situation is to renovate segments of the street when necessary. Gircle's street is very narrow, so a construction site at a street segment makes it impossible for citizens to pass that segment or even leave or enter it.

As a member of the Gircle Construction and Planning Commission (GCPC), you always know when one of the n street segments is closed or reopened. Naturally, the citizens expect you to tell them whether the trips they want to do are currently possible.

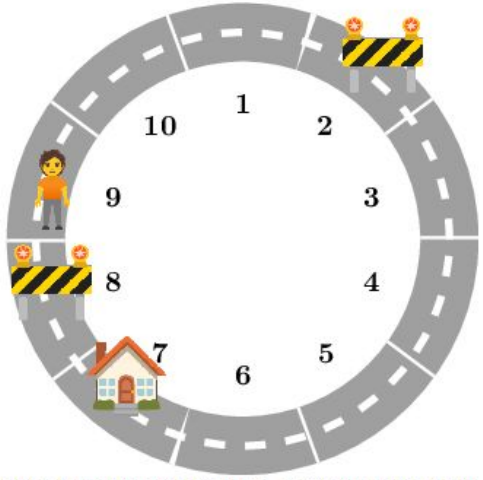


Figure 1. Depiction of the query "? 9 7" in the sample input.

Input

The input consists of:

- One line with two integers n ($2 \leq n \leq 10^5$) and q ($1 \leq q \leq 10^5$), the number of street segments and the number of events. No street segment is initially closed.
- q lines, each describing an event. Each event is described in one of the following ways:
  - "`- a`": Segment a ($1 \leq a \leq n$) is closed. It is guaranteed that segment a was open before.
  - "`+ a`": Segment a ($1 \leq a \leq n$) is reopened. It is guaranteed that segment a was closed before.
  - "`? a b`": A person asks you if it is possible to go from segment a to segment b ($1 \leq a, b \leq n$ and $a \neq b$).

Output

For each event of the form "`? a b`", print one line containing the word "`possible`", if it is possible to move from segment a to segment b, or "`impossible`" otherwise. If a or b are currently closed, the answer is "`impossible`".

Example

input Copy

```
10 12
? 1 5
- 2
- 8
? 9 2
? 9 8
? 9 7
? 6 7
? 3 7
? 1 9
? 9 1
+ 8
? 10 3
```

output Copy

```
possible
impossible
impossible
impossible
possible
possible
possible
possible
possible
```

Solution

- Build a segment tree on the count of the streets that are closed

Solution

- Build a segment tree on the count of the streets that are closed
- Whenever we're given that a street is opened or closed, we update it in our segment tree
    - Either set the street id to a value of 0 or 1 in our segment tree

Solution

- Build a segment tree on the count of the streets that are closed
- Whenever we're given that a street is opened or closed, we update it in our segment tree
    - Either set the street id to a value of 0 or 1 in our segment tree
- To see if a trip from ***A*** to ***B*** is possible, we can either go in decreasing or increasing order of street numbers

Solution

- Build a segment tree on the count of the streets that are closed
- Whenever we're given that a street is opened or closed, we update it in our segment tree
  - Either set the street id to a value of 0 or 1 in our segment tree
- To see if a trip from ***A*** to ***B*** is possible, we can either go in decreasing or increasing order of street numbers
- If either of these cases are true, the trip is possible
  - Sum of segment [***A***, ***B***] is zero, this represents going in increasing order
  - Sum of segment [***B***, ***n***] + segment [***0***, ***A***] == 0, this represents going in decreasing order

Example Problem - Inversion Counting

Given an array, the number of inversions is defined as the number of pairs of elements where the element to the left is larger than the element to the right

An array sorted in non-decreasing order will have zero inversions

We can count this in O(N^2) using brute force. Can we do better?

Example Problem - Inversion Counting

We can sort the array in descending order, and use a segment tree.

Example Problem - CSES Distinct Values

Distinct Values Queries

TASK | SUBMIT | RESULTS | STATISTICS | HACKING

Time limit: 1.00 s **Memory limit:** 512 MB

You are given an array of n integers and q queries of the form: how many distinct values are there in a range $[a, b]$?

Input

The first input line has two integers n and q: the array size and number of queries.

The next line has n integers $x_1, x_2, \ldots, x_n$: the array values.

Finally, there are q lines describing the queries. Each line has two integers a and b.

Output

For each query, print the number of distinct values in the range.

Attendance