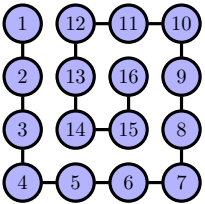
M1951 Spiral Sum

Alex Tung

30 June 2019

Statement

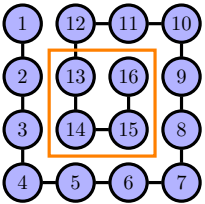
- Given a $N \times N$ grid with numbers



- Also given a sub-rectangle of the grid
  by the coordinates of two opposite corners
- Output the sum of numbers in the sub-rectangle mod 1000000007

Statement

- Given a $N \times N$ grid with numbers



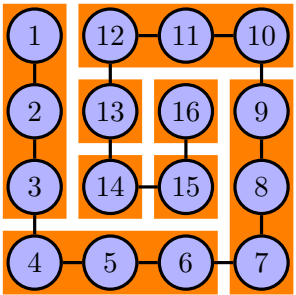
- Also given a sub-rectangle of the grid
  by the coordinates of two opposite corners
- Output the sum of numbers in the sub-rectangle mod 1000000007

Attempt 1

- Compute all the numbers in the grid
- Sum the required numbers one by one
- Time complexity: $O(N^2)$
- Expected score: 24

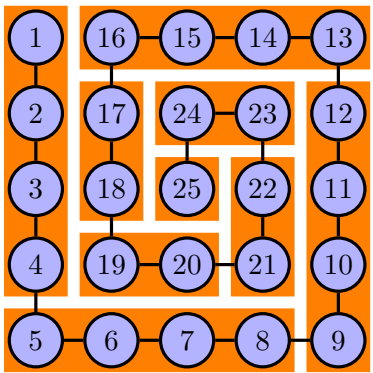
Attempt 2

- Consider O(N) strips



Attempt 2

- Consider O(N) strips



Attempt 2

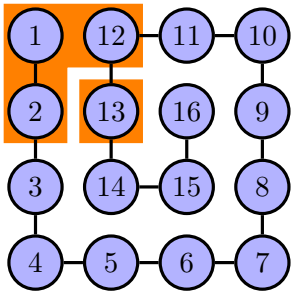
- For each strip, sum the required numbers in $O(1)$ by using arithmetic sum
- Time complexity: $O(N)$
- Expected score: 64

Attempt 3

- Cut horizontally and vertically in the middle to divide the grid into 4 regions
  - Handle the case for odd N carefully such that each number belongs to exactly one region
- For each region
  - it shares a corner with the original grid
    - e.g. $(1, 1)$ for the top left region
  - reduce the problem into 4 sub-problems such that the sub-rectangle also shares that corner by "2D prefix sum"
    - e.g. $\begin{cases} (r1', c1') = (1, 1) \\ (r2', c2') = (r1, c1), (r2, c2), (r1, c2), (r2, c1) \end{cases}$ for the top left region

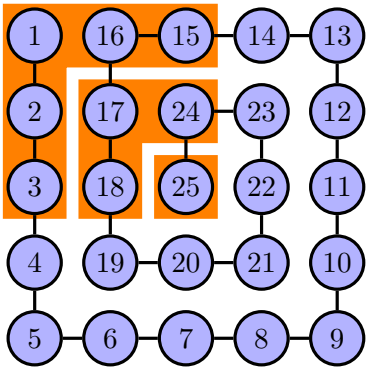
Attempt 3

- Consider O(N) L-pieces in each region



Attempt 3

- Consider O(N) L-pieces in each region



Attempt 3

- The intersection of each L-piece and the new sub-rectangle contains one (or two for the top left region) arithmetic sequence(s)
- Let a_i, b_i be the first term and the length of the arithmetic sequences respectively
- $(a_{i+1} - a_i)$ and (b_i) are arithmetic sequences themselves
  - This property relies on the fact that the new sub-rectangle shares a corner with the original grid and the division of the grid into 4 regions
- Then the sum of the arithmetic sums can be found by using the formula of $\sum_{k=1}^{n} k^p$

Attempt 3

- Instead of deriving tons of formulae, interpolation can also be used
- Using the fact: $\sum_{k=1}^{n} k^p$ is a degree $p+1$ polynomial of n with constants p, k
- We can deduce the original formulae are of almost degree 4
- Then we can use interpolation with the first 5 terms

Attempt 3

- Time complexity: $O(1)$
- Expected score: 100

M1952

Ian

tl;dr

Given Q queries, each could be one of the followings:

1. Insert a point(x, y) with weight w
2. Delete the point(x, y)
3. Find total weight of point P where mahattan distance of P and (x, y) is <= d
4. Find the farest point from (x, y)

Actually, I did not intended to add type 3 events, but I thought the problem set would be too easy if not.

Case 1-3

O(Q^2), just do what the problem says.

Finding Farest Manhattan Distance Point

manhattan distance of two point p, q = |p.x - q.x| + |p.y - q.y|

As we know, |a - b| = max(a - b, b - a), so the equation could be transformed to:

max(p.x - q.x, q.x - p.x) + max(p.y - q.y, q.y - p.y)

which is equals to the max of the following 4 equations:

p.x - q.x + p.y - q.y -> (p.x + p.y) + (-q.x + -q.y)

p.x - q.x + q.y - p.y -> (p.x - p.y) + (-q.x + q.y)

q.x - p.x + p.y - q.y -> (-p.x + p.y) + (q.x + -q.y)

q.x - p.x + q.y - p.y -> (-p.x + -p.y) + (q.x + q.y)

Case 4-6

In this subtask, there is no type 3 event and all type 4 events come after type 1, 2 events.

Let the query point be p. Using the previous 4 equations, for each of the following terms, we store the maximum:

(-q.x + -q.y)

(-q.x + q.y)

(q.x + -q.y)

(q.x + q.y)

And we enumerate which equations is the manhattan distance. (actually the one the maximum result would be it)

O(Q)

Case 10-13

Coordinate is small and the no. of combinations of that 4 terms is small too.

We could use some counting array to store the appearance of the terms.

O(Q * 400)

Case 14-19

Use 4 std::multisets or any other data structure to maintain the 4 terms.

Chebyshev Distance

Distance between two point = max(|p.x - q.x|, |p.y - q.y|)

Type 3, 4 events would be easier to handle, if the difference value become Chebyshev distance instead.

Type 4 events: maintain lowest, highest value of x, lowest, highest value of y.

Type 3 events: query sum of weights in (p.x - d, p.x + d, p.y - d, p.y + d).

Rotating the Plane

We could transform the Manhattan distance to Chebyshev distance!

Just rotate the plane by 45 degree.

(x, y) -> (x + y, x - y)

Now we are using Chebyshev distance instead.

Case 7-9

We can just CDQ divide and conquer to calculate the area sum.

Case 20-30

Use a 2D data structure (2D segment tree/BIT etc.), online maintaining area sum.

Since coordinate is large, you have to compress the segment tree.

Sightseeing Trail

30-6-2019

Jeremy Chow

Task Description

- Given an array consists of N integers and Q queries
- Each Query can be
  1. Add a triple (l_i, r_i, v_i)
  2. Delete a triple (l_i, r_i, v_i)
  3. Change a[x_i] to y_i
  4. Answer the total value in [l_i, r_i]
- Total value = sum of distinct value in [l_i, r_i] +

sum of v_j of triples where l_i <= l_j <= r_j <= r_i

Solution 1 (Case 1-4)

- Naive Solution
- For each type 1,2 query, add or delete the triple in a array
- For each type 4 query, go through [l_i, r_i] and the array of triples
- sum up the unique value and the valid triples
- Discretization / Map and long long to pass Case 3, 4

- Time complexity = O(NQ)
- Score = 16

Solution 2 (Case 5-8)

- Find the sum of unique value in [l_i, r_i]
- No update

- Mo’s algorithm (Square root decomposition on queries)
- Sort the type 4 queries by l_i / sqrt(N), then by r_i ascendingly
- Move the left and right pointer and maintain the sum of unique values

Solution 2 (Case 5-8)

```
Add(int x) {
        if (cnt[a[x]] == 0) res += a[x];
        cnt[a[x]]++;
}
Remove(int x) {
        if (cnt[a[x]] == 1) res -= a[x];
        cnt[a[x]]--;
}
```

Solution 2 (Case 5-8)

- Time complexity = O(N+QsqrtN)

- Discretization to pass Case 7-8 -> O(NlogN + QsqrtN)
- Poor implementation / Using Map -> O(QsqrtN * logN)

- Score = 32

Solution 3 (Case 9-12)

- Have type 1,2 operation
- Calculate the contribution of type 1,2 and type 3 separately
- Use Solution 2 to calucalute contribution of type 3

- Delete a triple (a, b, c)
- = Add a triple (a, b, -c)

Solution 3 (Case 9-12)

- Sort the type 1 and type 4 operation by r_i ascendingly

- Sweep line
- Process the queries by r_i ascendingly

- Add a triple (l_j, r_j, v_j), add v_j to index l_j if r_j <= r_i
- Answer = sum in [l_i, r_i]

Solution 3 (Case 9-12)

- Range Query
- Segment Tree / BIT

- Time complexity = O(NlogN + QsqrtN)
- Score = 48

Solution 4 (Case 13-16)

- ~~All type 4 operation is after type 1, 2, 3 operation~~
- Have update between queries
- Online

- CDQ Divide and Conquer
  - HKOI 2018 training camp D&C
  - https://assets.hkoi.org/training2018/dc.pdf

Solution 4 (Case 13-16)

```
CDQ(int l, r) {

    int mid = (l + r) / 2;

    CDQ(l, mid); CDQ(mid + 1, r);

    Calculate contribution of type 1,2 in [l, mid] for type 4 in [mid+1, r]

}
```

Solution 4 (Case 13-16)

- Calculate contribution of type 1,2 in [l, mid] for type 4 in [mid+1, r]

- All type 4 operation is after type 1, 2, 3 operation
- Use Solution 3

- Time complexity = $O(N + Q\log^2 N)$
- Score = 64

Solution 5 (Case 17-19)

- Have operation 3
- Any type of stamps can be found in atmost two attraction at any moment
    - Ocurrence of a_i <= 2

- When will it affect the answer?

Solution 5 (Case 17-19)

- Observation
- i <= j, a[i] = a[j]
- Answer -= a[i], if l_i <= i <= j <= r_i

- Looks like the type 1 operation!
- add triple (i, j, -a[i])

Solution 5 (Case 17-19)

- add some triple for the original array

- When encounter type 3 operation
  - Change a[x] to v
- add triple(x, i, a[x]) or triple(i, x, a[x]), if a[i] = a[x] and i != x
  - Remove contribution
- add triple(x, j, -v) or triple(j, x, -v), if a[j] = v and j != x
  - Add contribution

Solution 5 (Case 17-19)

- Type 3 operation is tranformed into type 1 operation
- Store indice by vector / array / pair

- Do CDQ just like Solution 4
- Time complexity = $O(N + Q\log^2 N)$
- Score = 76

Full solution

- Have operation 3
- Can’t calculate contribution directly in CDQ

- Transfrom operation 3 to operation 1!

Full solution

- Let the original array [1, 3, 5, 3, 3, 5]
- Consider the sum of unique value in [2, 6]

- As each unique value is only counted for once, we only add the leftmost one
- [1, 3, 5, 3, 3, 5] = 3 + 5
- = sum of value in [2, 6] - sum of redundant value
- [1, 3, 5, 3, 3, 5] = (3 + 5 + 3 + 3 + 5) - (3 + 3 + 5) = 8

Full solution

- When will a number become redundant ?
- [1, 3, 5, 3, 3, 5], l <= 2 <= 4 <= r
- [1, 3, 5, 3, 3, 5], l <= 4 <= 5 <= r
- [1, 3, 5, 3, 3, 5], l <= 2 <= 5 <= r
- triples{(2, 4, -3), (4, 5, -3), (3, 6, -5)}

- if a[i] = x, add a triple(j, i, -x) if there exist j s.t. a[j] = x and j < i

Full solution

- Sum of unique value in [l, r] = sum of value in [l, r] + contribution of triples

- Type 3 operation can tranform to type 1 in a similar manner
- Add fews triples to modify the change

Full solution

- [1, 3, 5, 3, 3, 5] -> triples{(2, 4, -3), (4, 5, -3), (3, 6, -5)}
- Change a[4] to 5
- [1, 3, 5, 5, 3, 5]
- Add (2, 4, 3), (4, 5, 3), (2, 5, -3)
  - Remove the contribution of i
- Add (3, 4, -5), (4, 6, -5), (3, 6, 5)
  - Add the contribution of i
- One type 3 can be transform to at most Six type 1

Full Solution

1. Add some triples based on the originally array
2. For type 1, 2, just add the triples
3. For type 3
    3.1. Find the largest u s.t. u < x_i and a[u] = a[x_i]
    3.2. Find the smallest v s.t. v > x_i and a[v] = a[x_i]
    3.3. Add some triples by using u and v
    3.4. a[x_i] = v_i
    3.5. Find the largest u s.t. u < x_i and a[u] = a[x_i]
    3.6. Find the smallest v s.t. v > x_i and a[v] = a[x_i]
    3.7. Add some triples by using u and v
4. For type 4, add up the sum of value in [l_i, r_i] first
5. Do solution 5

Full Solution

- Find the largest u s.t. u < x_i and a[u] = a[x_i]
- Find the smallest v s.t. v > x_i and a[v] = a[x_i]
- Can be done with std :: set

- Add up the sum in [l_i, r_i]
- Can be done with segment tree / BIT
- Time complexity = $O(N\log N + Q\log^2 N)$

Propagation

Jason – 30/6/2019

Problem Statement

- Given a rooted tree with N nodes, each with its own value a_i
- Answer a number of queries:
  - For a node x_i, what is the sum of a_i of all t_i^{th} level descendants of x_i

- The queries are given online that you need to compute them one by one
  - Each x_i depends on the previous answer

Case 1 – 4 (~8%)

- The graph is a chain
- The t_i^{th} level descendant of node x_i is just node $(x_i + t_i)$

- Just output the value of the node $(x_i + t_i)$

Case 5-8 (~16%)

- $Q, N \leq 200$

- We can check whether a node is a k-th level descendant of another node one by one by finding its k-th level ancestor
- Finding this takes O(N^2) time (or O(N log N))

- Time complexity: O(QN^2) or O(QN log N)

Case 9-14 (~28%)

- $Q, N \leq 2000$
- We can perform DFS starting from a node to find out all k-th level descendants of that node at once
- Time complexity: O(QN)

Case 15-20 (~40%)

- $Q, N \leq 200000$ but all $t_i \leq 10$
- We can precompute the answers for all nodes

- Perform DFS starting from the root
- When a node is visited, add its a_i to the corresponding answers of its 1^{st} to 10^{th} ancestors
- Then all queries can be answered directly

Case 15-20 (~40%)

```
procedure dfs(node u, int depth):
    stack[depth] = u
    for i from 1 to 10:
        answer[stack[depth – i]][i] += a[u]
    for all children v of u:
        dfs(v, depth + 1)
```

- Time complexity: O(*N* + *Q*)

Case 21-32 (~64%)

- $Q, N \leq 200000$ and all a_i are divisible by N
- Since $x_i = (y_i + ans_{i-1}) \; mod \; N$, $x_i = y_i$ now
- The queries can be done offline

Case 21-32 (~64%)

- Find the pre-order of the tree and build a segment tree on it storing the range sum
- Consider the layers one by one in descending order of the depths
- For each layer,
  - Insert the values a_i of all nodes at that layer into the segment tree
  - Then answer the queries about that layer by finding the range sum using the segment tree
  - Undo the changes before handling the next layer

- Time complexity: $O((Q + N) \log N)$

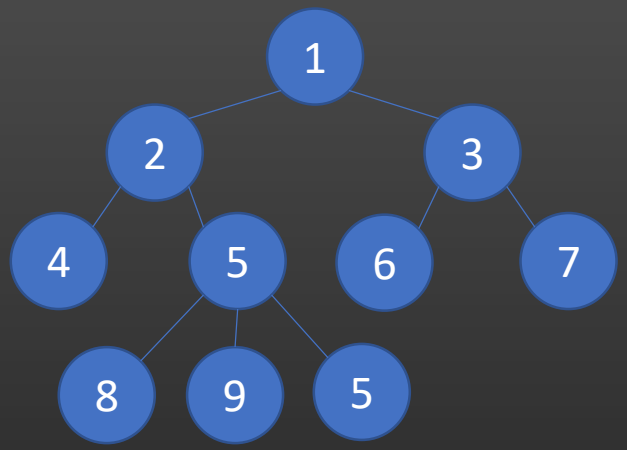
Case 33-50 (100%)

- $Q, N \leq 200000$, online queries now

- Perform DFS starting on the root and store the starting time, finishing time and depth of each node
- And for each layer, store all nodes at that layer in ascending order of their starting time
  - Push the node into the corresponding vector during DFS traversal

Case 33-50 (100%)

- To answer a query, find all nodes at the layer $t_i + depth[x_i]$ that has the starting time $st[j]$ that $st[x_i] \leq st[j] \leq ft[x_i]$
- Can perform binary search on the corresponding vector to find the range of the required nodes and get the sum of a_i of the range
  - The sum can be found easily by precomputing the prefix sums of the vectors
- Time Complexity: O(N + Q log N)



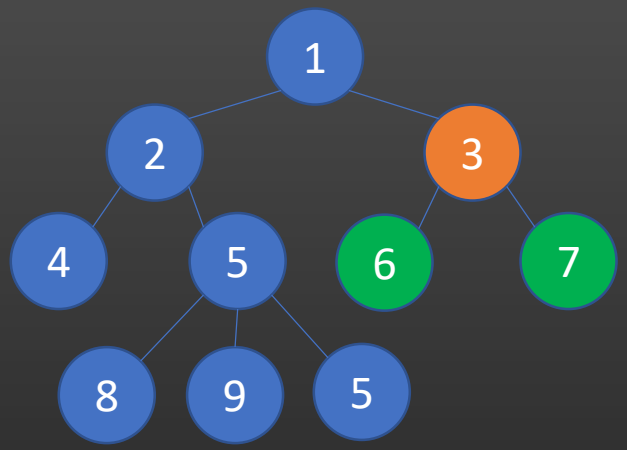
Node	1	2	3	4	5	6	7	8	9	10
a_i	1	2	3	4	5	6	7	8	9	5
Starting Time	1	2	14	3	5	15	17	6	8	10
Finishing Time	20	13	19	4	12	16	18	7	9	11

Depth=0	
Starting Time	1
a_i	1
Prefix Sum	1

Depth=1		
Starting Time	2	14
a_i	2	3
Prefix Sum	2	5

Depth=2				
Starting Time	3	5	15	17
a_i	4	5	6	7
Prefix Sum	4	9	15	22

Depth=3			
Starting Time	6	8	10
a_i	8	9	5
Prefix Sum	8	17	22



Node	1	2	3	4	5	6	7	8	9	10
a_i	1	2	3	4	5	6	7	8	9	5
Starting Time	1	2	14	3	5	15	17	6	8	10
Finishing Time	20	13	19	4	12	16	18	7	9	11

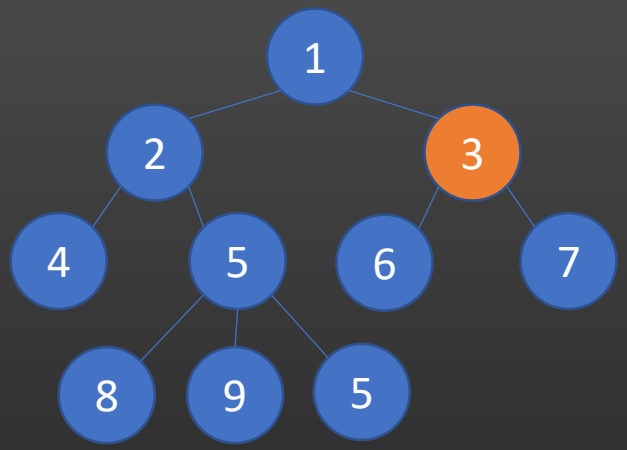
x = 3, t = 1:
Find the nodes at depth=2
with starting time within 14~19
Answer = 22 – 9 = 13

Depth=0	
Starting Time	1
a_i	1
Prefix Sum	1

Depth=1		
Starting Time	2	14
a_i	2	3
Prefix Sum	2	5

Depth=2				
Starting Time	3	5	15	17
a_i	4	5	6	7
Prefix Sum	4	9	15	22

Depth=3			
Starting Time	6	8	10
a_i	8	9	5
Prefix Sum	8	17	22



Node	1	2	3	4	5	6	7	8	9	10
a_i	1	2	3	4	5	6	7	8	9	5
Starting Time	1	2	14	3	5	15	17	6	8	10
Finishing Time	20	13	19	4	12	16	18	7	9	11

x = 3, t = 2:
Find the nodes at depth=3
with starting time within 14~19
No such nodes, answer = 0

Depth=0	
Starting Time	1
a_i	1
Prefix Sum	1

Depth=1		
Starting Time	2	14
a_i	2	3
Prefix Sum	2	5

Depth=2				
Starting Time	3	5	15	17
a_i	4	5	6	7
Prefix Sum	4	9	15	22

Depth=3			
Starting Time	6	8	10
a_i	8	9	5
Prefix Sum	8	17	22