**Question 3 Room Colors**

**Divyansh Mathur,210357**

**Algorithm: Maintain Binary Indexed Trees (BITs) for Room Colors and Update Times**

Input:

* n: the number of rooms
* queries: a list of queries of the form (l, r, c), where l and r are room indices, and c is the color to update

Output:

* Answer: an array of size n, containing the final colors of all rooms

Procedure:

1. Calculate N = 2^ceil(log2(n)) to determine the size of the binary tree.
2. Initialize two arrays of size 2\*N-1:
   * color (initialize with the original color and dummy nodes)
   * time (initialize to -1)
3. Initialize a variable t to 0 to keep track of the time of updates.
4. For each query (l, r, c) in queries, do the following:
   * Update the color and time of nodes from l to r as follows:
     + Set color[N-1+l] = color[N-1+r] = c
     + Set time[N-1+l] = time[N-1+r] = t
5. For each query (l, r, c), increment t by 1 (i.e., t++).
6. Initialize an Answer array of size n.
7. For each index i from 0 to n-1, do the following:
   * Initialize max\_time as time[N-1+i]
   * Set Answer[i] = color[N-1+i]
   * Recursively traverse towards the root:
     + If time[parent] > max\_time, update max\_time to time[parent and set Answer[i] = color[parent].
8. Return the Answer array as the final colors of all rooms.

***Pseudocode:***

Function initializeBITs(n):

N = 2^ceil(log2(n))

Create empty arrays color and time of size 2\*N-1

Initialize each element of color from N-1 to N-1+n-1 with the original color

Initialize each element of time to -1

Function update(n, l, r, c, t):

N = 2^ceil(log2(n))

u = N - 1 + l

v = N - 1 + r

color[u] = c

color[v] = c

time[u] = t

time[v] = t

while parent(u) != parent(v):

if u < v:

if isLeftChild(u):

color[u + 1] = c

time[u + 1] = t

u = parent(u

if not isLeftChild(v):

color[v - 1] = c

time[v - 1] = t

v = parent(v

Increment t by 1 (t++)

Function query(n):

N = 2^ceil(log2(n))

Create an empty array Answer of size n

For i from 0 to n-1:

max\_time = time[N - 1 + i]

Answer[i] = color[N - 1 + i]

node = N - 1 + i

while node > 0:

parent = parent(node)

if time[parent] > max\_time:

max\_time = time[parent]

Answer[i] = color[parent]

node = parent

return Answer

Top of Form