S211 - Skyscraperhenge

Ethen Yuen (ethening)

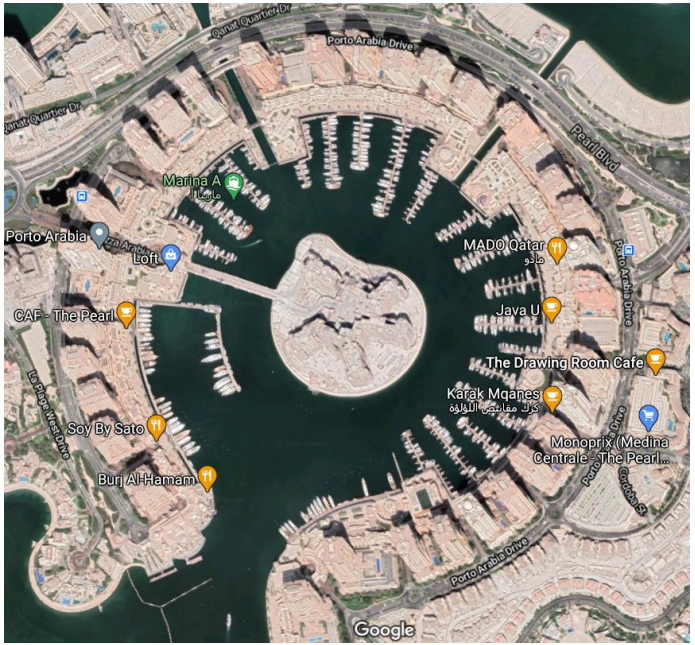
香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Background

Problem Idea by ethening

Preparation by ethening, trashcan, percywtc

Pictures by microtony



The Pearl, Doha, Qatar in Google Map satellite view

(Satisfies the constraints of Subtask 2)

Problem Restatement

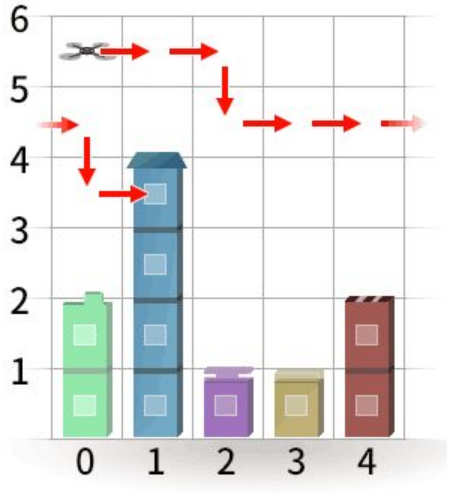
Given a cycle of **N** buildings with height **A[0..N-1]**.

A drone, starting at position **0**, height **(H + 0.5)** follows string **S** of length **K** repeatedly

- **'N'**: position **x** → **(x + 1) % N**
- **'D'**: height **y** → **y - 1**

At any point, (y < A[x]) → **CRASH**

Determine: **"FOREVER"** / **"TOP x"** / **"SIDE x"**



5 5 2 4 1 1 2 4 NNDN	SIDE 1

Statistics

0 points	9	+ 2	+ 1	+ 0	= 12
19 points	4	+ 0	+ 0	+ 0	= 4
21 points	7	+ 13	+ 5	+ 0	= 25
40 points	0	+ 4	+ 6	+ 3	= 13
69 points	0	+ 0	+ 2	+ 1	= 3
100 points	0	+ 0	+ 1	+ 4	= 5

First solved by **dbsgame** at **1h 11m 48s**

SUBTASK

For all cases:

$3 \le N \le 100000$

$A_0 \le H \le 2 \times 10^9$

$1 \le A_i \le 2 \times 10^9$

$1 \le K \le 100000$

	Points	Constraints
1	21	$3 \le N \le 100$ $A_0 \le H \le 100$ $1 \le A_i \le 100$ $1 \le K \le 100$ It is guaranteed that the drone will crash and stop at some point
2	19	All A_i are equal In string S, no command `N` appears after a command `D`
3	29	In string S, at least half of the commands are `N`
4	31	No additional constraints

Subtask 1

Subtask 1 (21%): N, H, K, A[i] ≤ 100. There must be a crash at some point.

- No need to handle “FOREVER” cases.
- Observe that N, H, K are **small**.
- Pure **simulation** might be able to pass.

Subtask 1

PSEUDOCODE

```
curr_x = 0, curr_y = H
While (TRUE)
  For i = 0 .. K - 1
    If (S[i] is ‘D’) curr_y = curr_y - 1
    Else if (S[i] is ‘N’) curr_x = (curr_x + 1) % N
    If (curr_y < A[curr_x])
        If (S[i] is ‘D’) Output “TOP curr_x”
        Else if (S[i] is ‘N’) Output “SIDE curr_x”
        Stop the program
```

Subtask 1

- Pure **simulation** should be able to pass.
- Case 1: No “D” in S
  - Since there must be a crash (at height H) → Crash within first N steps
- Case 2: Exist “D” in S
  - H iterations of string S must crash the drone into ground

Score: 21

Time Complexity**:** *O(max(N, HK))*

Determination of “FOREVER” case

- Case 1: No “D” in S
  - The drone remains at height H
    - Case 1a: Exist x such that H < A[x] → **Crash (“SIDE x”)**
    - Case 1b: Otherwise → **“FOREVER”**
- Case 2: Exist “D” in S
  - H iterations of string S must crash the drone into ground
- We can focus only on **Case 2** from now on

Subtask 2

Subtask 2 (19%): All A[i] are equal. No “N” appears after “D”.

- Suppose A[i] = a, the drone crashes when its **height < a**
- In S, let number of “N” = **dx**, number of “D” = **dy**.
  (dx can be 0, dy > 0 after handling “FOREVER”)

Subtask 2

Subtask 2 (19%): All A[i] are equal. No “N” appears after “D”.

- In S, number of “N” = **dx**, number of “D” = **dy**.
- We can treat S as two big steps:
  - `x → (x + dx) % N`
  - `y → y - dy`
- As all A[i] are equal, it is certain that there are no crashes during the horizontal movement.

Subtask 2

PSEUDOCODE

```
curr_x = 0, curr_y = H
While (TRUE)
  curr_x = (curr_x + dx) % N
  curr_y = curr_y - dy
  if (curr_y < a)
    Output “TOP curr_x”
    Stop the program
```

Time Complexity: *O(H/dy)*

Too slow to pass this subtask

Subtask 2

- Observe that the above algorithm can be optimized by **calculation**:
  - Starting height of drone: H
  - Each iteration: -dy
  - Stop when height < a
  - The while loop runs for `ceil( (H - a + 1) / dy )` times.

Subtask 2

PSEUDOCODE

```
curr_x = 0
r = ceil((H - a + 1) / dy)

curr_x = (curr_x + r * dx) % N

Output “TOP curr_x”
```

As the number is large, *long long* (64-bit integer) needs to be used

Score: 19 (Cumulative: 40)

Time Complexity: *O(1)*

Crucial Observations

- Suppose max(A[i]) = M, the drone will not crash when height ≥ M.

This seems obvious and useless at first glance; how does it help us in solving this task?

- If height of drone < M, the drone must crash into a building after at most N commands “N”.

How does this observation, when combined with the above, brings us closer to the full solution?

Crucial Observations

- Suppose max(A[i]) = M, the drone will not crash when height ≥ M.
- If height of drone < M, the drone must crash into a building after at most N commands “N”.

We could design an algorithm that consists of 2 parts:

1. Bring down the drone to just slightly above M.
2. Simulate the commands repeatedly to determine the position of crashing.

Subtask 2 (Calculation) + Subtask 1 (Simulation)

Subtask 3

Subtask 3 (29%): At least half of S are “N”.

- We shall leave the constraints of this subtask for discussion later.

Subtask 3

Part 1: Skip by Calculation

- As no crashing occurs when height ≥ M, the order of “N” and “D” in S does not matter, so only the numbers of each are important **(dx and dy)**.

- Similar to subtask 2, calculate the number of iterations of string S **before reaching M** and determine the position of drone after the skip.
- Number of iterations = **floor((H - M) / dy)**

Time Complexity: *O(1)*

Subtask 3

Part 2: Simulation step by step

- Continue after the skip, just perform the same algorithm as subtask 1.
- 1 more iteration of string must get the drone's height < M
- Then, after at most N commands "N", the drone must crash into a building.

Time Complexity: *O(KN / dx)*

Subtask 3

Part 1: Skip by Calculation

PSEUDOCODE

```
curr_x = 0, curr_y = H
r = floor((H - M) / dy)

curr_x = (curr_x + r * dx) % N
curr_y = curr_y - r * dy
```

Part 2: Simulation step by step

PSEUDOCODE

(refer to Subtask 1)

Subtask 3

Part 1: Skip by Calculation

Time Complexity: *O(1)*

Part 2: Simulation step by step

Time Complexity: *O(KN / dx)*

Since in subtask 3, at least half of S are “N” → **Time Complexity: *O(N)***

Score: 29 (Cumulative: 69)

Full Solution

Subtask 4 (31%): No additional constraints.

- What is left to optimize?

In subtask 3, part 2 (Simulation step by step)

- Time Complexity: *O(KN / dx)*
- When **dx is small**, the algorithm is too slow to pass.

Full Solution

- Is there a way to transform string S into some form that is more favourable for our algorithm?

- Remember in subtask 2: we treat continuous “N” and “D” into 2 big steps?
- As long as there is no crash during the big step, this combined step acts the same as individual steps.

- Let’s merge continuous “D” into a big step.

Full Solution

- Let's just merge continuous "D" into a big step.
  - The outcome ("TOP x") **is the same** for the drone crashes after whichever individual "D" step. The only thing matters is whether it crashes during this continuous segment.
  - The string after merging is similar to the string we get for subtask 3. "N" takes up ≥ half of S (or maybe slightly less than half of S).
- The steps can be merged at the start by a simple loop

Corner Case

- If string S **consists of only "D"** (dx = 0), the answer must be "TOP 0".
- Since the simulation part stops after at most N commands "N", it will **never stop until hitting the floor** for this case → Too slow

- Some contestants fails to full this task because they did not handle this case :(
- This case is hinted by sample 4.

Full Solution

Merge commands "D" *O(K)*

Handle dy = 0 ("FOREVER" or crash at height H) *O(N)*

Handle dx = 0 *O(1)*

Part 1: Skip by Calculation *O(1)*

Part 2: Simulation by Merged steps *O(N)*

Score: 100

Time Complexity: $O(K + N)$

Alternative Full Solution

(Solution by **mtyeung1** during contest)

Handle the corner cases first (dx = 0, dy = 0).

For each building, if at some time the drone crashes into that building, all time onwards when the drone revisit that zone it would crash.

It is possible to use binary search to determine that first crashing time.

At the end, compare the crashing time of each building. The one having the minimum crashing time is the answer we need.

Alternative Full Solution

Binary search for the **crashing time of drone** (how many “N” is executed) for each building.

Consider the case of crashing at **SIDE** first. For building x, we only consider time in the form of **(cN + x)**.

> To determine crashing, perform another binary search: how many “D”s has been executed when the (cN + x)-th “N” is being executed.
>
> *(Height before moving to the zone required)*
>
> If (H - (number of “D”) < A[x]) then CRASH

Alternative Full Solution

Consider the case of crashing at **TOP**. For building x, we only consider time in the form of **(cN + x + 1)**.

To determine crashing, perform another binary search: how many “D”s has been executed when the (cN + x + 1)-th “N” is being executed.

(Height before moving onto the next zone)

If (H - (number of “D”) < A[x]) then CRASH

Score: 100

Time Complexity: *O(N log(HK) log(K))*