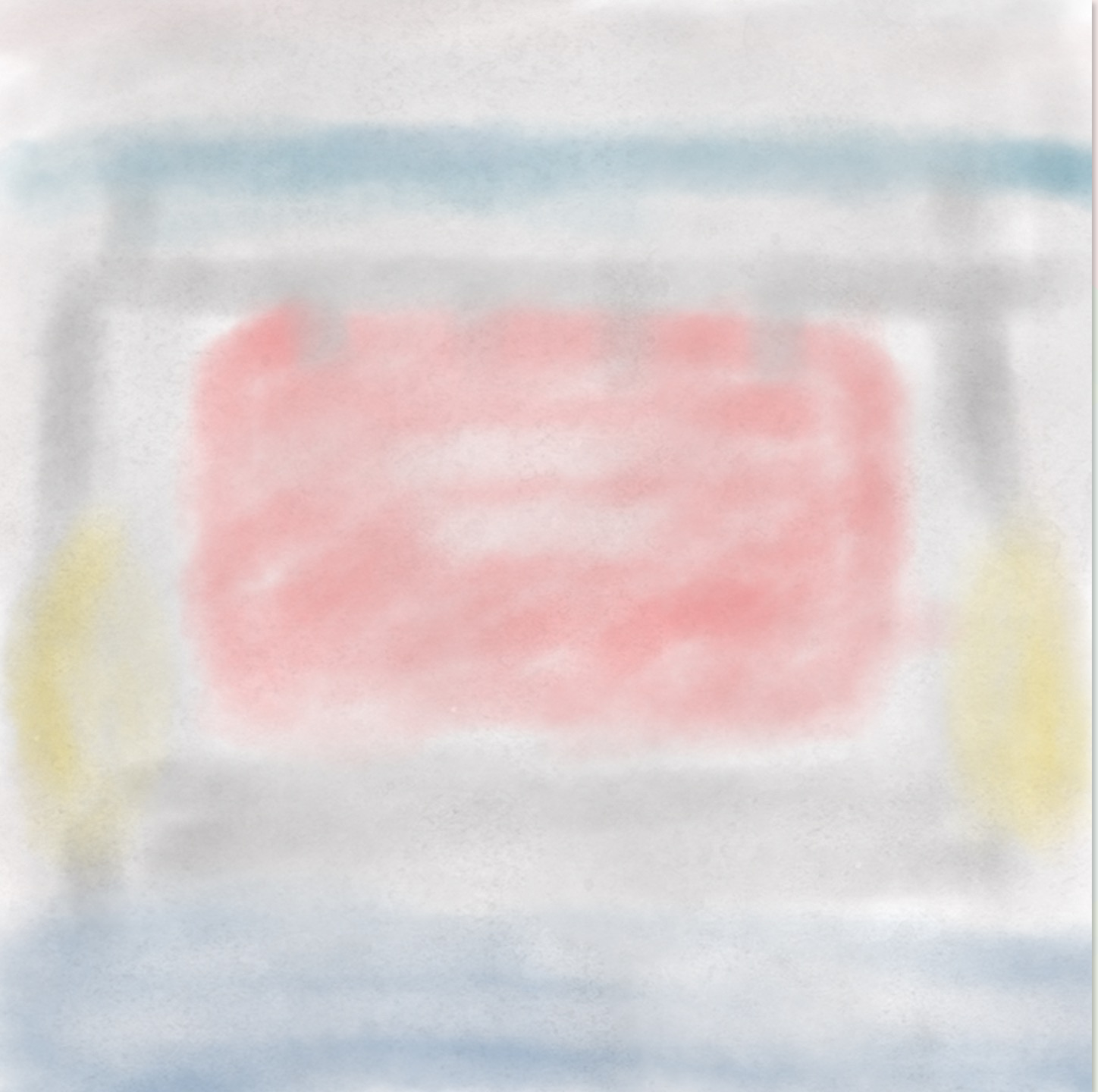
TWGSS OI Team

2022/23 iHCC INTER-HOUSE CODING COMPETITION

EDITORIAL

6th July 2023

Jack Wong Daniel Hsieh



TH23

1

Mosaic Colour

Story

Fun fact:

EVERY MTR station have different colours!

For the full colour list, visit

https://zh.wikipedia.org/zh-hk/Template:港鐵車站顏色

Problem Statement

There is a string S of N characters.

Put letters in `*` positions such that all adjacent letters are different.

Can only use the first K alphabets.

If possible, output the resultant string.

Otherwise, output `AC So Exciting`.

Subtask 1

S consists of `*` only

Case 1 $K = 1$

We can only use `A` in the answer.

- If $N = 1$, just output `A`.
- Otherwise, it is impossible to get a valid answer as it can only be `AAAA...`. Therefore, output `AC So Exciting`.

Subtask 1

S consists of `*` only

Case 2 $K \geq 2$

We can at least use both `A` and `B` in the answer.

It is always possible!

We can just output `ABABAB...` or `BABABA...`, which are both valid.

Expected Score	9	Cumulative	9

Subtask 2

$1 \leq N \leq K \leq 26$

As the number of alphabets that we can use is not less than the length of the string, it is always possible, and we can just output an unused alphabet in the string for each `*`.

Open a `bool` array and check whether each alphabet is used.

For each `*`, replace it with one of the unused alphabet, then set the state of that alphabet to used.

Expected Score	8	Cumulative	17

Subtask 3

$K = 2$

We can only use `A` and `B` in the answer.

From Subtask 1 Case 2, we know that there are only 2 ways for this.

- `ABABAB...`
- `BABABA...`

We can just check whether all the already-present characters fit at least one of the above arrangements.

Output any suitable one if yes. Otherwise, output `AC So Exciting`.

Expected Score	14	Cumulative	31

Full Solution

There are only 3 cases:

- $K = 1$ Impossible when $N \geq 2$; $N = 1$ covered in Subtask 1
- $K = 2$ Subtask 3
- $K \geq 3$

Full Solution

For $K \geq 3$, it is always possible!

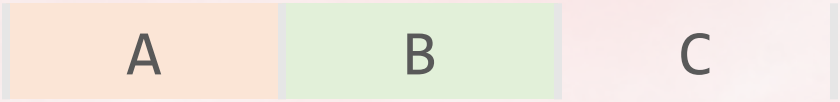
In fact, we can just use `A`, `B` and `C` to fill in the whole string.

For each `*`, we can just fill it in using the first alphabet which is unused from the one on its left and right.

Full Solution

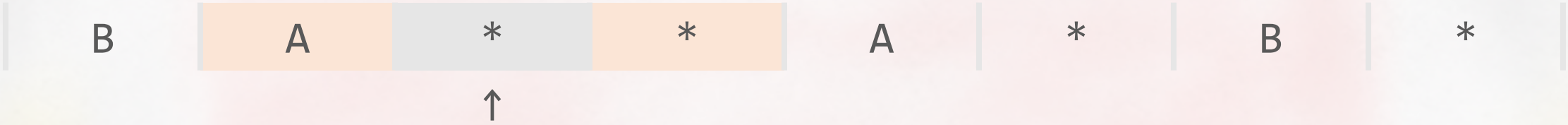
$K \geq 3$ Implementation

Chosen

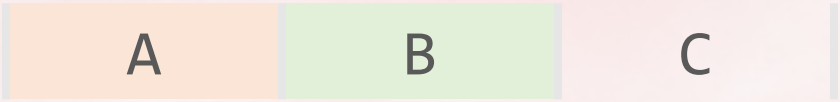


Full Solution

$K \geq 3$ Implementation

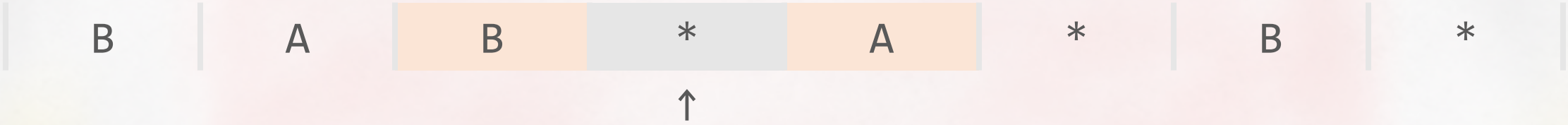


Chosen

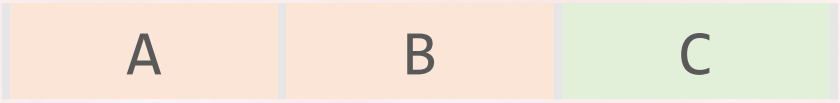


Full Solution

$K \geq 3$ Implementation



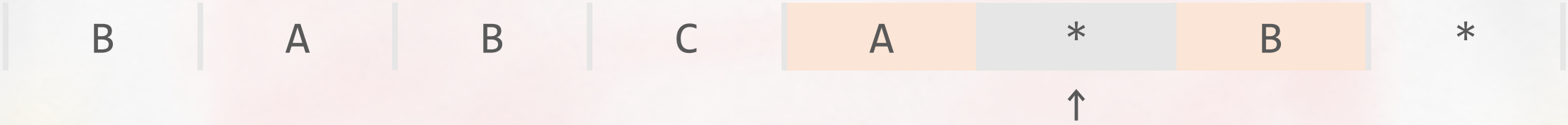
Chosen



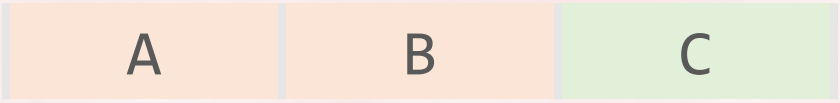
Full Solution

$K \geq 3$

Implementation



Chosen



Full Solution

$K \geq 3$

Implementation

Chosen

Full Solution

$K \geq 3$ Implementation

B	A	B	C	A	C	B	A

Done!

Expected Score	50	Cumulative	50

TH23

2

Elective Voting

Problem Statement

Given the votes of each topic, determine the number of sessions for each topic using Hare Quota:

- Hare quota $\lambda = \frac{\text{number of votes } V}{\text{number of sessions } Q}$ (λ is always integer)
- For each topic, for each λ votes it got, a session is allocated to that topic. If a total of k sessions is allocated, $k\lambda$ votes would be subtracted from the valid vote of that topic.
- For the remaining session(s), they are assigned to topics following the descending order remaining valid votes, one by one. In other words, a topic with more remaining valid votes would have priority over a topic with a less remaining valid votes.
- In case of topics having the same remaining valid votes but there is not enough remaining sessions, extra sessions would be held for those topics.

Subtask 1

$1 \leq V \leq 10^9$
$Q = 1$

Only 1 session!

Just allocate the session to a topic with the maximum number of vote!

Simple!

Subtask 1

$1 \leq V \leq 10^9$
$Q = 1$

Only 1 session!

~~Just allocate the session to a topic with the maximum number of vote!~~

No!

What if more than one topics have the maximum number of vote?

→ Allocate a session to ALL topics with the maximum number of vote!

Expected Score	8	Cumulative	8

Subtask 2

$1 \leq V \leq 10^9$

A_i is divisible by λ

Simple!

Just divide each A_i by λ!

That's it!

Expected Score	6	Cumulative	14

Subtask 3

$1 \leq V \leq 2.2 \times 10^6$
$1 \leq N \leq 82$
$1 \leq Q \leq 35$

Weird constraints?

I wonder where these numbers came from…

Subtask 3

$1 \leq V \leq 2.2 \times 10^6$
$1 \leq N \leq 82$
$1 \leq Q \leq 35$

- Calculate λ
- Allocate the “full” sessions
- Use any method to order the topics according to the number of remaining votes they got
- From high to low, allocate the topics with extra sessions
- If more than one topics have the same highest remaining votes, give a session to each of these topics

Expected Score	21	Cumulative	35

Subtask 4

$1 \leq N \leq 5000$

The constraints limiting V is gone!

Highest possible $V = 5000 \times 10^9 = 5 \times 10^{12}$

May I proudly introduce… `long long`!

Expected Score	28	Cumulative	42

Subtask 5

The situation where multiple topics having the same remaining valid votes but there is not enough remaining sessions would not happen

We can ignore the annoying rule of distributing the extra sessions!

But there is no longer constraints limiting N!

We would get TLE if we use O(N²) method to sort the remaining number of votes as maximum $N^2 = (2\times10^5)^2 = 4\times10^{10}$.

We have to use O(NlogN) method (e.g. merge sort) or just use C++ `sort()` instead.

Expected Score	17	Cumulative	53

Full Solution

Combine Subtask 4 solution with Subtask 5 solution.

- Use `long long`
- Use C++ `sort()`
- Handle the annoying situation of distributing the extra sessions

Done!

Expected Score	70	Cumulative	70

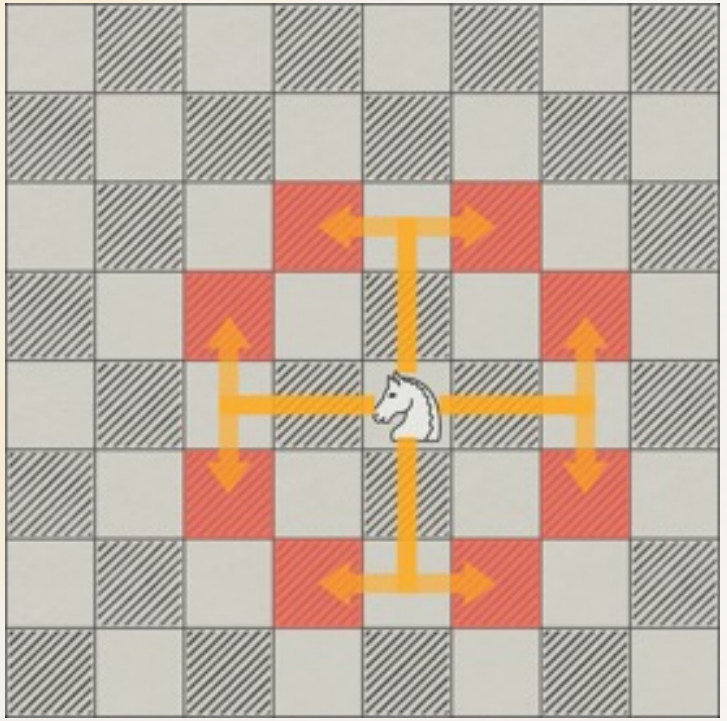
TH23

3

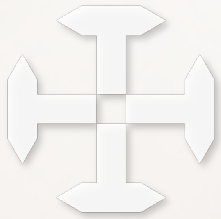
Good Knight

Problem Statement

- Given an 8×8 Chess board, a Knight and a King
- The Knight can move from opposite corners of an 1×2 rectangle to another (like normal chess) in 1 unit of time



- The Knight has to capture the king at time T **while keep moving**
- Determine whether it is possible



Subtask 1

$0 \leq T \leq 1$

Case 1 $T = 0$

The Knight can capture the King if and only if they occupy the same position.

Subtask 1

$0 \leq T \leq 1$

Case 2 $T = 1$

The Knight can capture the King if and only if the Knight can reach the King's position in one move.

- Method 1 Enumerate all 8 directions

  $(+1, +2), (+1, -2), (+2, +1), (+2, -1),$

  $(-1, +2), (-1, -2), (-2, +1), (-2, -1)$

- Method 2 Note that Knight can move from (x_1, y_1) to (x_2, y_2) only if

  $(x_1 - x_2)^2 + (y_1 - y_2)^2 = 5$

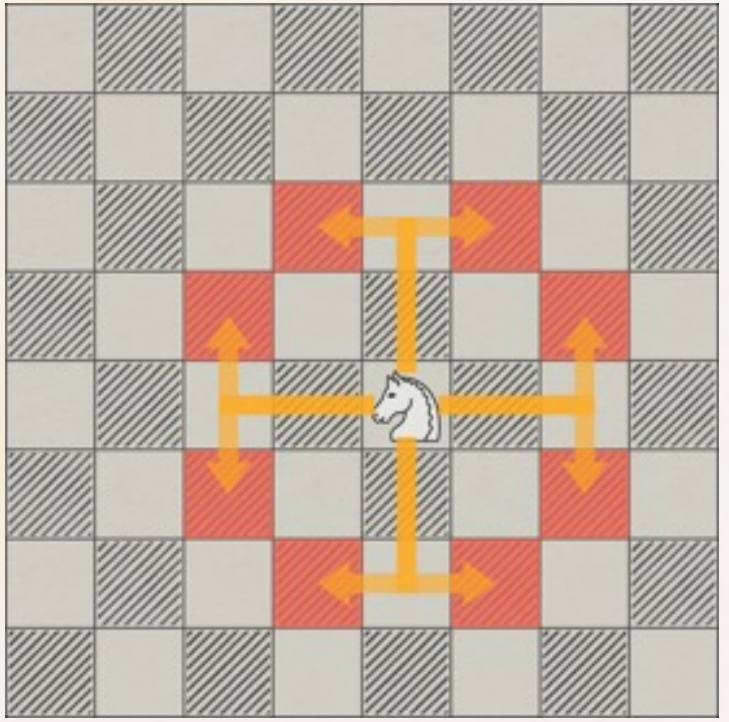
Expected Score	13	Cumulative	13

Subtask 2

Observation 1

For each move, the Knight can only move to a cell in **different colour** than the current one.

- White cell → Black cell
- Black cell → White cell



Subtask 2

$(a_x, a_y) = (b_x, b_y)$

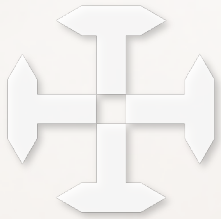
Case 1 T is even

- The Knight can repeatedly move between cells in cycle like

  A → B → A → B → A → …

Case 2 T is odd

- As the Knight can only move to opposite coloured cell, the final cell colour must be different from the initial cell colour.



Subtask 2

$(a_x, a_y) = (b_x, b_y)$

When T is even, output `Good Knight`.

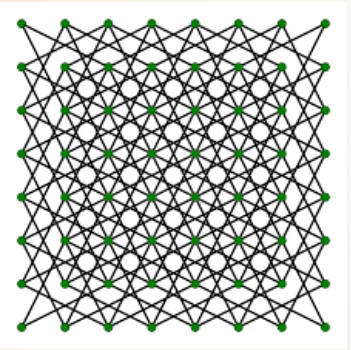
When T is odd, output `God Save The King`.

Extremely easy to code!

Expected Score	7	Cumulative	20

Subtask 3

$0 \leq T \leq 10$



Graph!

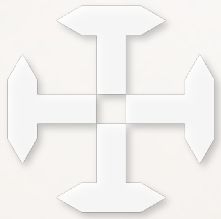
We can treat the cells as nodes and Knight moves as edges.

Run DFS to check if it is possible to go from (a_x, a_y) to (b_x, b_y) in exactly T Knight moves.

Time Complexity: $O(8^{10}) \approx O(10^9)$

Might sounds a lot, but it works as on the border of the board, there are fewer possible Knight movements.

Expected Score	37	Cumulative	44



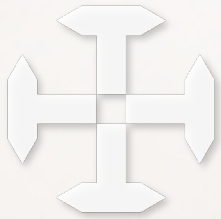
Full Solution

As T can be as large as 10^9, it is too large to simulate!

Observation 2

Only the shortest possible Knight moves is required.

- If the parity of T does not match the colouring, then it is impossible.
- Otherwise, we can move the Knight from A to B, then B to A, "wasting" 2 moves each time.



Full Solution

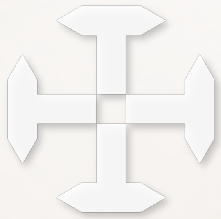
Therefore, the answer is `Good Knight` when…

- $T \geq$ minimum number of Knight moves to the King

AND

- $T \bmod 2 = (a_x + a_y + b_x + b_y) \bmod 2$ → Correct chessboard colouring

Otherwise, the answer is `God Save the King`.



Full Solution

You can use graph algorithms (e.g. BFS, Dijkstra) to find the shortest distance in a graph.

Alternatively, as there are only $8 \times 8 = 64$ nodes, some heuristic algorithm might also work:

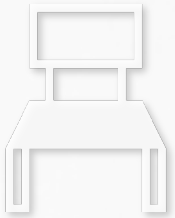
- Let `D[x][y]` := the minimum number of Knight moves from (a_x, a_y) to (x, y)
- `D[w_x][w_y] = 0`, and set all other values to `+inf`
- Randomly select a cell (x_1, y_1) and consider all cells (x_2, y_2) reachable from (x_1, y_1)
- We can assign `D[x_2][y_2] = min(D[x_2][y_2], D[x_1][y_1] + 1)`
- Repeat the process for many times, and `D[x][y]` will eventually be correct

Expected Score	80	Cumulative	80

TH23

4

Personalised Seating

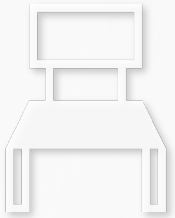


Problem Statement

Given N integers $A_1, A_2, \ldots, A_N$.

You have to pair them up (with at most one remains).

So that the sum of differences of each pair is either **maximised** (Subtask 1) or **minimised** (Subtask 2-4).



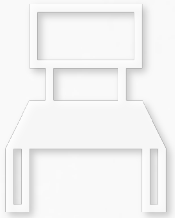
Subtask 1

$c = \mathrm{R}$ **Maximise** the sum of differences

Observation 1

The order of `A[]` does not matter.

- We can first sort the array with C++ `sort()` in O(NlogN)
- From now, we can suppose `A[1] ≤ A[2] ≤ … ≤ A[N]`



Subtask 1

$c = \mathrm{R}$ **Maximise** the sum of differences

Observation 2

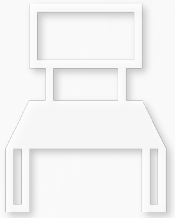
To **maximise** the sum of differences, we should pair `A[1]` with `A[N]`, `A[2]` with `A[N-1]`, etc.

- Proof leave as exercise for readers ☺

We may use for-loop to loop through all pairs and add their difference up.

Time Complexity: O(NlogN)

Expected Score	21	Cumulative	21



Subtask 2

$c =$ F

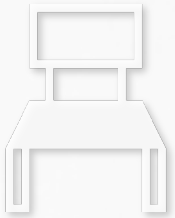
Minimise the sum of differences

N is even

Observation 3

To **minimise** the sum of differences, we should always pair neighbour elements of A[ ].

Otherwise, we can always lower the sum by swapping or changing pairs.



Subtask 2

$c = \mathrm{F}$ **Minimise** the sum of differences

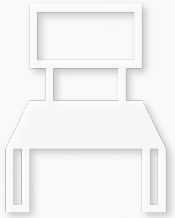
N is even

When N is even, the pairing is unique.

We can just simply loop through all the pairs and add their difference up.

Time Complexity: O(NlogN)

Expected Score	11	Cumulative	32



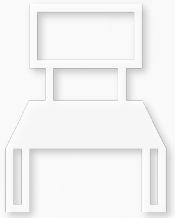
Subtask 3

$c =$ F **Minimise** the sum of differences

$1 \leq N \leq 5000$

Case 1 N is even

- Apply Subtask 2 solution



Subtask 3

$c =$ `F` **Minimise** the sum of differences

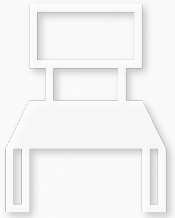
$1 \leq N \leq 5000$

Case 2 N is odd

- Loop i from 1 to N, remove `A[i]` form the array
- Then apply Subtask 2 solution to each of the edited arrays
- Find the minimum answer among them

Time Complexity: $O(N^2)$

Expected Score	23	Cumulative	55



Subtask 4

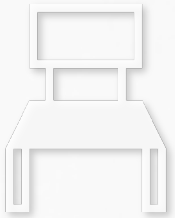
$c =$ `F` **Minimise** the sum of differences

For odd N, we have to do pre-calculation!

Total number of pairs $= \frac{N-1}{2}$

- Let `L[k]` be the sum of differences in pairs $(1,2),(3,4),\ldots,(N-2,N-1)$
- Let `R[k]` be the sum of differences in pairs $(N,N-1),(N-2,N-3),\ldots,(3,2)$

`ans = min(L[k] + R[N/2-k])` for `0 ≤ k ≤ N/2`



Subtask 4

$c =$ `F` **Minimise** the sum of differences

We can use the idea of **partial sum** to calculate the array `L[]` and `R[]`.

For `k = 0, 1, 2, …, N/2`:

- `L[k] = L[k-1] + (a[2k] – a[2k - 1])`
- `R[k] = R[k-1] + (a[N–(2k-1)] – a[N–2k])`

Time Complexity: O(NlogN)

Expected Score	79	Cumulative	100

TH23

5

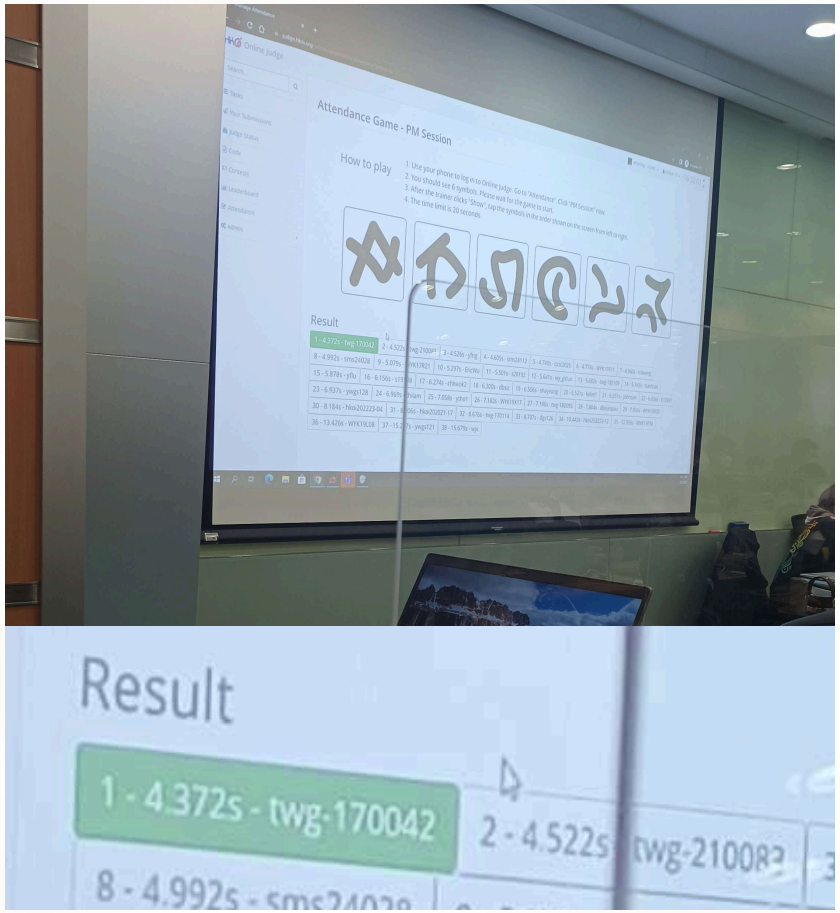
Attendance Game

Story

The HKOI attendance game is fun!

If you want to experience it, you have to…

- Work hard
- Get a medal in next year's HKOI
- Join the training!



Problem Statement

Given a correct string S of length M.

For participants numbered 1 to N,

they will perform a total of Q operations to their answer A:

- c x — Append x copies of c to the end of A
- `delete` x — Delete the last x characters in A
- `clear` — A is cleared
- `submit` — **Finish** if $A = S$, otherwise clear A

Target: Get the finish order of the participants.

Subtask 1

S consists of `A` only

$c =$ `A`

Just use an integer to keep track of the length in the answers.

For participant p, let the length of his/her answer be `A[p]`:

- c x — `A[p] += x`
- `delete` x — `A[p] -= x`
- `clear` — `A[p] = 0`
- `submit` — `if (A[p] == M)` **finish**; `else` `A[p] = 0`

Subtask 1

S consists of `A` only

$c =$ `A`

When participant p finishes the game, and `ok[p] == false`

- push p into the output `vector`
- set `ok[p]` to `true`

Finally, output the size and the content of the `vector` in order.

Subtask 1

S consists of `A` only

$c =$ `A`

Still not done?

Let's see what would be theoretically the largest integer that will appear in your code…

$M \times Q = 10^5 \times (3 \times 10^5) = 3 \times 10^{10}$

Uh oh… `long long` it is.

Subtask 1

S consists of `A` only

$c =$ `A`

Add a `long long`,

Results go strong!

Die on `long long`,

Many marks gone!

Result

Subtask	Verdict	Score
1	Wrong Answer	0 /15
2	Accepted	37 /37
3	Wrong Answer	0 /30
4	Accepted	22 /22
5	Wrong Answer	0 /26

Source Code C++20

```cpp
#include <bits/stdc++.h>
using namespace std;

vector<pair<char,int>> st;
vector<vector<pair<char,int>>> ans;
vector<int> len,done;
set<int> check;
string s;
bool can = true;
int N,M,Q,p,x;

int main(){
    cin >> N >> M >> Q;
    cin >> s;
    st.push_back({s[0],1});
    for(int i = 1;i<s.size();i++){
        if(s[i]==s[i-1]){
            st[st.size()-1].second++;
        }else{
            st.push_back({s[i],1});
        }
```

Expected Score	15	Cumulative	15

Subtask 2

$1 \leq N, M, Q \leq 2000$

Pure, naïve simulation.

For participant p, let his/her answer string be `A[p]`:

- c x — `A[p] += c` — for `x` times
- `delete` x — `A[p].pop_back()` — for `x` times
- `clear` — `A[p] = ""`
- `submit` — `if (A[p] == S)` **finish**; `else` `A[p] = ""`

Expected Score	37	Cumulative	52

Subtask 4

$x = 1$ for all valid operations

i.e. only one letter will be added or deleted for each respective operation

No more simulation!

Instead, for each participant, we can keep track of his/her…

- Length of **correct prefix** in the answer `A[p]`
- Length of **wrong suffix** in the answer `B[p]`

Subtask 4

$x = 1$ for all valid operations

i.e. only one letter will be added or deleted for each respective operation

For each operation:

- c 1
```
if (A[p] < M && c == S[A[p]] && B[p] == 0)      A[p]++
else                                            B[p]++
```
- delete 1
```
if (B[p] > 0)     B[p]--
else              A[p]--
```
- clear
```
A[p] = B[p] = 0
```
- submit
```
if (A[p] == M && B[p] == 0)   finish
else                          A[p] = B[p] = 0
```

Expected Score	22	Cumulative	74

Full Solution

Similar to Subtask 4.

Keep track of each participant's…

- Length of **correct prefix** in the answer `A[p]`
- Length of **wrong suffix** in the answer `B[p]`

As $x \geq 1$, we have to **pre-calculate** the **consecutive "equal" strings** as adding x letters works only when the string has a lot of consecutive equal characters.

Full Solution

For each the i^{th} character in the S, let `R[i]` be the remaining length of the current **consecutive "equal" string**.

i	0	1	2	3	4	5	6	7	8	9
S[i]	A	A	A	B	A	A	C	C	C	C
R[i]	3	2	1	1	2	1	4	3	2	1

Full Solution

For each operation:

- c x

```
if (A[p] < M && c == S[A[p]] && B[p] == 0)
    B[p] += max(0, x-min(R[A[p]], M-A[p]))
    A[p] += min(R[A[p]], M-A[p], x)
else
    B[p] += x
```

- delete x

```
A[p] -= max(0, x-B[p])
B[p] -= min(B[p], x)
```

- clear

```
A[p] = B[p] = 0
```

- submit

```
if (A[p] == M && B[p] == 0)   finish
else                          A[p] = B[p] = 0
```

Expected Score	130	Cumulative	130

Subtask 3

There is no `delete` or `clear` operations

A safety net for the full solution in case you get these two operations wrong.

Expected Score	30

TH23

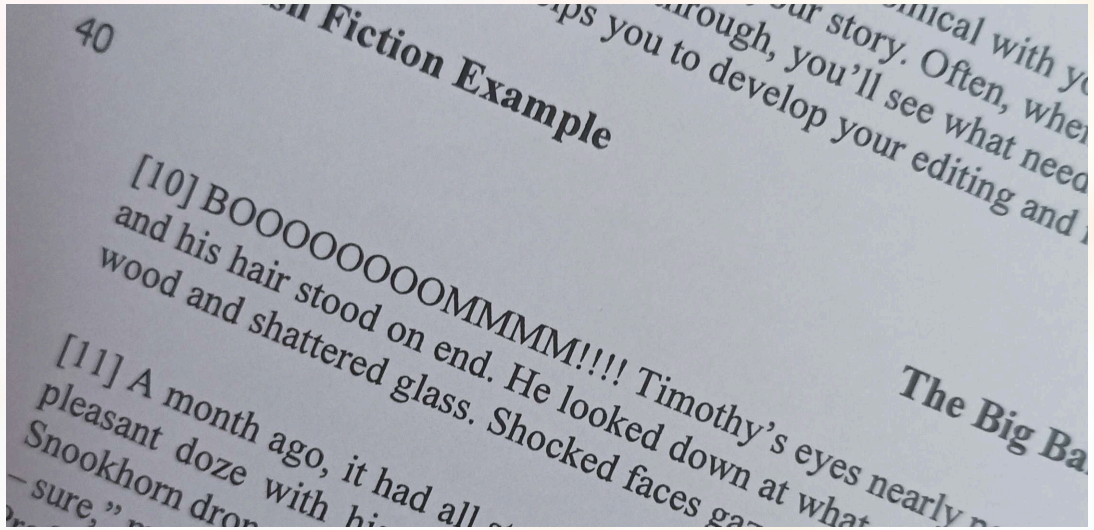
6

Dangerous Solution

Story

You will sooner or later watch the famous ***flash fiction "The Big Bang"*** featuring Timothy in the *Super Science Experiments for Eager Youngsters* competition in the 2023 DSE past paper!

BOOOOOOOOMMMM!!!!


40

Fiction Example

[10] BOOOOOOOOMMMM!!!! Timothy's eyes nearly
and his hair stood on end. He looked down at what
wood and shattered glass. Shocked faces

The Big Ba

[11] A month ago, it had all
pleasant doze with
Snookhorn dro
– sure,"

Problem Statement

Given N liquids.

Each are either acidic (type A) or basic (type B).

You need to group all test tubes into two groups so that each group consists of test tubes containing the same type of liquids.

You can carry out trials by putting liquid in some test tubes together.

- If you add c_A units of type A liquid and c_B units of type B liquid, the explosion strength is $\min(c_A, c_B)$ (similar to that in real life!)
- **Units of liquid poured from each test tube each trial ≤ 1024**
- **Minimise the number of trials K**

K = 999

Pour solution in test tube 1 into container 1.

Then for each $i = 2, 3, \ldots, N$, add liquid in test tube 1 and test tube i together.

- If explosion occurs, then they have different type.

  → Pour solution in test tube i into container 2

- Otherwise, they have the same type.

  → Pour solution in test tube i into container 1

Expected Score	30.078	20.052	%

K = 500

For each of $1 \leq k \leq \left\lfloor \frac{N-1}{2} \right\rfloor$,

- `addLiquid(1, 3)`
- `addLiquid(2k, 1)`
- `addLiquid(2k+1, 2)`

For different results, we know exactly the types of test tube $2k$ and $2k+1$ relative to test tube 1!

If N is even, we have to consider test tube N separately.

Expected Score	**69.15**	46.1	%

Assume test tube 1 is type A:

Explosion Strength	Type of test tube $2k$	Type of test tube $2k+1$
0	A	A
1	B	A
2	A	B
3	B	B

K = 100

We can use binary number to generalise the idea!

Note that **units of liquid poured from each test tube** ≤ 1024.

Therefore, we can test 10 test tubes each time! For each valid k,

- `addLiquid(1, 1024)`
- `addLiquid(10k-9, 1)`
- `addLiquid(10k-8, 2)`

  ⋮
- `addLiquid(10k, 512)`

We can know the types of liquids contained in them consider the bits of the explosion strength.

K = 100

Assume test tube 1 is type A:

Explosion Strength		Type of test tube				
Decimal	Binary	$10k-9$	$10k-8$		$10k-1$	$10k$
0	0000000000	A	A		A	A
1	0000000001	B	A		A	A
2	0000000010	A	B		A	A
3	0000000011	B	B		A	A
⋮				…	⋮	
1021	1111111101	B	A		B	B
1022	1111111110	A	B		B	B
1023	1111111111	B	B		B	B

Expected Score	100.5	67	%

K = 91

Can we optimize further?

Yes!

We can test 11 test tubes at a time.

- Noting if we add two **known** and **same type** solution of of volume 1024 together initially, we can test with $1, 2, 4, 8, \ldots, 1024$.
- Note that first three test tubes must have at least two being the same.

Expected Score	127.5	85	%

Full Solution K = 88

Group of 12!

We add ***three*** **known** and **same type** (assume as type A) solution of of volume 1024 together initially, we can test with $1, 2, 4, 8, \ldots, 512, 1024, 1024$.

We can tell the content of the first 10 test tubes (up to 512) by idea mentioned before.

Full Solution K = 88

For the 11th and the 12th test tube (both 1024),

- If they have the same type, we can immediately distinguish them:

  If strength < 1024 → both type A

  If strength ≥ 2048 → both type B

- Otherwise, we still know they have different type.

  Test the 11th test tube again later

  → Deducing the 12th test tube using the result

Therefore, on average, we can tell (11+12)÷2 = 11.5 test tubes per trial.

Expected Score	150	100	%

TH23

7

Left Alright

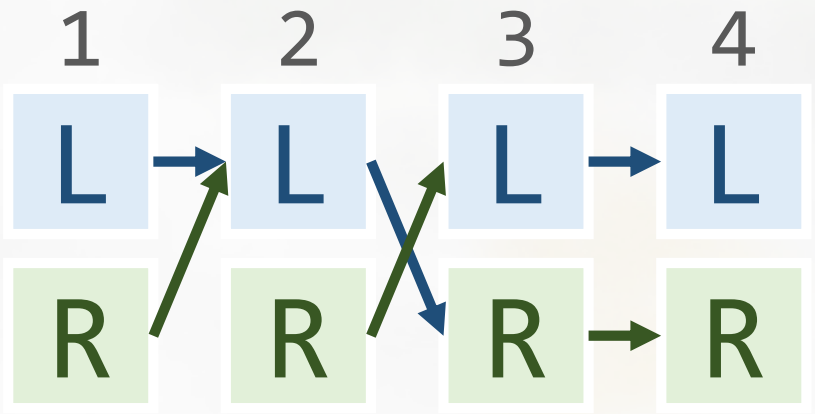
Problem Statement

There are N friends.

Each friend have preferences:

- Answer direction `L` / `R` when heard `L`
- Answer direction `L` / `R` when heard `R`

Example:



Problem Statement

Two Types of Operations:

- **Update** (1) Friend f change his/her preference
- **Game** (2) Given L and R,
  find the final direction for friends $L, L+1, \ldots, R$
  if friend L heard direction d

Subtask 1

$1 \leq N, Q \leq 5000$

For each query, simply simulate what happened, from left to right.

That's it.

Time Complexity: O(NQ)

Expected Score	34	Cumulative	34

Subtask 2

$x = 1$ for every **“Game”** operation

Any **“Update”** operations are before any **“Game”** operations

Simply deal with all the updates naïvely.

Note that there are totally N possibilities for **“Game”** operations as $x = 1$.

We can **pre-compute** the answers start with `L` or `R` from $y = 1$ to $y = N$

Time Complexity: O(N+Q)

Expected Score	17	Cumulative	51

Subtask 3

$L_i \neq R_i$ for $1 \leq i \leq N$

$l \neq r$ for every **“Update”** operation

There are at most 10 **“Update”** operations

Note that we can treat `R L` as $+1$ (swapped) and `L R` as 0 (not swapped).

Therefore, we only have to know the number of times swapped the directions within the range of the **“Game”** operations using **partial sum**.

We can re-compute the partial sum after each updates. (Note that you still have to compute it at the beginning.)

Time Complexity: O(Q+11N)

Expected Score	40	Cumulative	91

Subtask 4

$L_i \neq R_i$ for $1 \leq i \leq N$

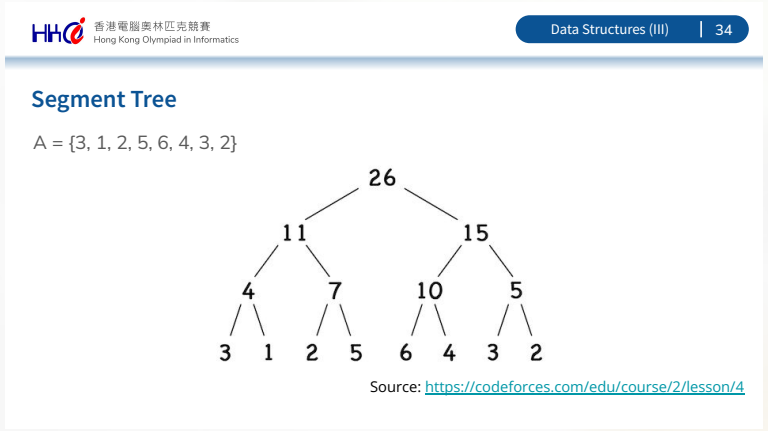
$l \neq r$ for every **“Update”** operation

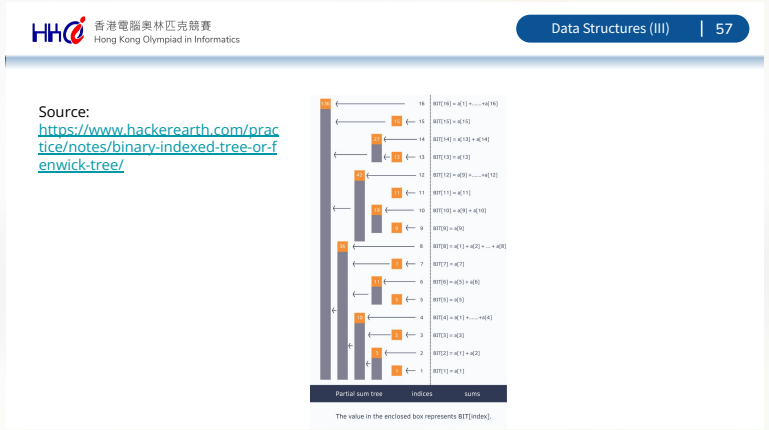
How can we maintain the sum of ranges quickly?

We can use either **segment tree** or **binary indexed tree** to maintain them! (You can refer to HKOI Training Materials for more information.)

Time Complexity: O((N+Q) lgN)

Expected Score	59	Cumulative	110





Subtask 5

There are no **“Update”** operations

Search the **last** `L L` or `R R` position in the **“Game”** operation range:

- If such position does not exist, then the Subtask 4 idea works
- Otherwise, we can simply ignore anything before that `L L` or `R R`
- We starts here, and since no more `L L` nor `R R` exists after this position, we can again use Subtask 4 idea to solve it

Use `set` to store all the positions of `L L` or `R R`

Time Complexity: O((N+Q) lgN)

Expected Score	44	Cumulative	154

Full Solution

Combine the idea in Subtask 4 and Subtask 5.

Subtask 5 Search the **last** `L L` or `R R` position in the **"Game"** range.

Use `set` to maintain all the positions of `L L` or `R R`.

Subtask 4 Maintain the **number of "swaps"** in ranges **quickly**

Use **segment tree** or **binary-indexed tree**

Time Complexity: O((N+Q) lgN)

Expected Score	180	Cumulative	180

Alternative Solution

You can use **matrix** to solve this problem also, which work for any number of items. (Not limiting to `L` and `R`!)

Let M be a matrix that $M_{ij} = 1$ if the person see item i, he/she say item j.

Otherwise, $M_{ij} = 0$.

Then we can use a **segment tree** to find the **product of the matrices over any ranges**.

Time Complexity: O((N+Q) K^3 lgN) ($K = 2$ in this problem)

Expected Score	180	Cumulative	180

TH23

8

Destined Gate

Story

The story is referencing to *Steins;Gate*.

Problem Statement

Given a weighted directed graph.

You may walk along the edges as you want.

In some vertexes, you can jump against the direction of the edges, but the sum of weights of the path must be $\leq T$.

Find the **minimum number of jumps** required to go from a certain vertex to the destination.

Subtask 1

$1 \leq Q \leq 10$

$c_i = 0$ for $1 \leq i \leq N - K$

No Time Leap Machines.

Just **DFS** from the given event, and see if can reach the **Destined Gate**.

- If the **Destined Gate** is reachable, output `0`.
- Otherwise, if it is not possible to go to the **Destined Gate**, output `-1`.

Time Complexity: O(Q(N+M))

Expected Score	18	Cumulative	18

Subtask 2

$1 \leq N, M, Q \leq 300$

$c_i = 1$ for $1 \leq i \leq N - K$

Consider a graph, which for all possible **time leaps** into the past, we build a **directed** edge of cost 1, and the normal edge with cost 0.

The minimum number of **time leaps** needed is the minimum cost from the given event to the **Destined Gate**, `-1` if unreachable.

We can find the minimum cost by using **BFS** / **Dijkstra** algorithm.

Time Complexity: $O(Q(N^2+M))$ or $O(Q(N^2+M) \lg N)$

Expected Score	26	Cumulative	44

Subtask 3

$1 \leq N, M, Q \leq 5000$

Observation 1

It is always better to go to the **farthest** past.

- Can go back to the future anyways
- Can be found in O(N) by repeatedly go back whenever possible

Subtask 3

$1 \leq N, M, Q \leq 5000$

Build edge only if the **Time Leap Machine** is available at event k, i.e. $c_k = 1$.

This reduces the total number of edges to $N + M$.

Simply Run **BFS / Dijkstra** Q times.

Time complexity: O(Q(N+M)) or O(Q(N+M) lg(N+M))

Expected Score	62	Cumulative	80

Subtask 4-5

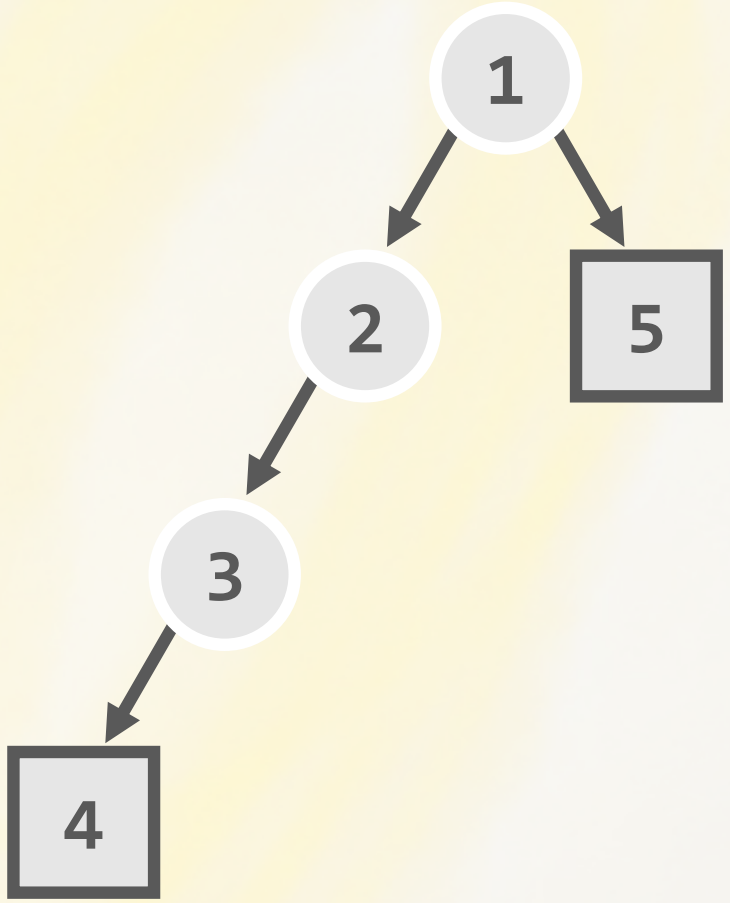
$K = 2$

$M = N - 1$

$u_i = i, v_i = i + 1$ for $1 \leq i \leq N - 2$

$u_{N-1} = 1, v_{N-1} = N$

“Chain” graph with $K = 2$



Subtask 4

"Chain" graph with $K = 2$

$c_i = 1$ for $1 \leq i \leq N - K$

Case 1 $e_j = N - 1$

- The answer is always `0`

Case 2 $e_j = N$

- We have to reach node 1
- We may repeatedly use **Time Leap Machines**

Subtask 4

"Chain" graph with $K = 2$

$c_i = 1$ for $1 \leq i \leq N - K$

Case 2 $e_j = N$

- Precompute the farthest past reachable for every node in O(N)
  Move the farthest past whenever the distance too large
- **Dynamic Programming** from the root to the leaf by **DFS**
  Maintain the minimum cost along the way

Time Complexity: O(N)

Expected Score	29	Cumulative	109

Subtask 5

"Chain" graph with $K = 2$

Change the order of DP calculation

- Compute all $c_i = 1$ first in the first DFS
- Then compute for all $c_i = 0$ again using DFS

More difficult!

Time Complexity: O(N)

Expected Score	51	Cumulative	131

Subtask 4-5

We may also use **Binary Lifting**.

Consider 2^K **Time Leaps** for $K = \lfloor \lg N \rfloor, \lfloor \lg N - 1 \rfloor, \ldots, 0$.

- Can be precomputed in O(NlgN)
- Let `anc[v][K]` be the oldest **spacetime point** can be reached after 2^K **Time Leaps**, then `anc[v][K + 1] = anc[anc[v][K]][K]`

For each K, we try to time leap whenever 2^K **time leaps** are not sufficient, i.e. need to get to even older spacetime point.

We jump once more if necessary.

Expected Score	51	Cumulative	131

Subtask 6

$c_i = 0$ for $1 \leq i \leq N - K$

Either `0` or `-1`, depending on whether can go to the **Destined Gate**.

Observation 2

Suppose we add all the **endings** to the node connected to them.

The answer is `0` if and only if the **Destined Gate** exists in the subtree of the **event**.

How can we check?

Subtask 6

Flatten the Tree using **Euler Tour**!

It becomes a range query problem.

We can maintain sets of all occurring indexes for each possible **Gate**.

- The answer is `0` if and only if the current range corresponding to the subtree contains any of the indexes

Time Complexity: O((N+Q) lgN)

Expected Score	39	Cumulative	170

Subtask 7

$1 \leq K \leq 10$

We can consider each possible **Gate** separately.

By **DFS** on the reverse graph, we can compute the list of nodes that can go to a specific **Gate**.

Minimum **Time Leaps** needed to jump to such a node.

Can be solved using Subtask 5 idea.

Time Complexity: O(NK)

Expected Score	80	Cumulative	199

Full Solution

Combine ideas in:

- Subtask 6 Flatten Tree
- Subtask 4-5 Binary Lifting

Precompute the farthest past reachable for every node in O(NlgN)

Consider the destination after 2^K **Time Leaps** for $K = \lfloor \lg N \rfloor, \lfloor \lg N - 1 \rfloor, \ldots, 0$.

- If the destination can reach the gate, we do not **Time Leap**
- Otherwise, we need to **Time Leap**

We might require an extra **Time Leap** to reach the gate.

Full Solution

Each checking as in Subtask 6 is O(lgN), and we jump O(lgN) times.

Time Complexity: $O(Q\lg^2 N)$

Expected Score	240	Cumulative	240

TWGSS
OI Team

2022/23

iHCC INTER-HOUSE CODING COMPETITION

THANK YOU!

End of Editorial