TWGSS
OI Team

2022/23 Mini Contest II

Editorial

(Predicted) Cut-offs

Award	TC232**1** Rotating Dice	TC232**2** Can't Wake Syndrome	TC232**3** Silver Legacy	**Total**
Gold	100	66	100	**266**
Silver	69	36	51	**206**
Bronze	40	24	20	**84**
HM	17	5	9	**31**

Rotating Dice

TC2321

Statistics

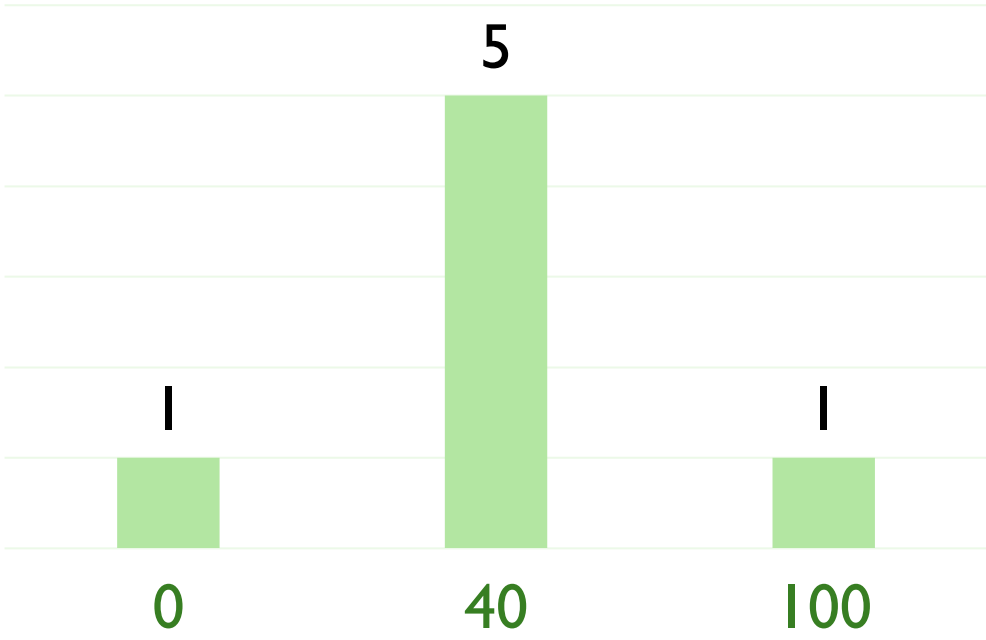
Total Score	100

Attempts	7

Subtask Performances

Subtask	Score	Solves
1	7	6
2	10	6
3	23	6
4	29	1
5	31	1

Score Distribution



First Solve	DuncanNG	1:13:11

Problem Statement

- Given an initial state of a dice.
- Do some rotation.
- Output the final state of the dice.

- Easy to understand!
- But obviously (slightly) tedious!

- Inspiration from Rubik's cube.
- Simulation: https://alg.cubing.net/?puzzle=1x1x1.

Subtask 1

- S contains `F` only
- Note sample 1.

- `a[] = {F, B, R, L, U, D}`
- We can just repeatedly update the faces of the dice
  - `a'[2] = a[4]`
  - `a'[3] = a[5]`
  - `a'[4] = a[3]`
  - `a'[5] = a[2]`

- Expected Score: 7 (Cumulative: 7)

Subtask 2

- S contains `F` and `B` only
- Note sample 1 **and 2**.

- We can just repeatedly update the faces of the dice
  - For `F`: Same as Subtask 1
  - For `B`:
    - `a'[2] = a[5]`
    - `a'[3] = a[4]`
    - `a'[4] = a[2]`
    - `a'[5] = a[3]`
- Alternatively, note that `B` = `FFF`

- Expected Score: 17 (Cumulative: 17)

Subtask 3

- S contains capital letters only

- We can just repeatedly update the faces of the dice:
  - `a[?] = a[?], … *6`
  - Enumeration!
- Alternatively, notice that `B` = `FFF` etc.
  - `a[?] = a[?], … *3` only!
- Use sample to test your solution!

- Expected Score: 40 (Cumulative: 40)

Subtask 4

- S might contain `()n`.

- As Length of $S \leq 20$, just "unpack" the brackets!
- While True:
  - Find the "minimal" brackets
  - Replace it by several copies of itself
- Then just employ solution to subtask 3.

- Expected Score: 69 (Cumulative: 69)

Full Solution

- Time complexity grows exponentially! `(((((((()9)9)9)9)9)9)9)`
- Obviously, we cannot simply unpack the rotations.

- Define x, y be any rotation. Note we can calculate x^n and xy.
- Main Idea: We may treat `(x)n` as exponentiation, x^n. Instead of exploding the whole thing up, we merge the "simple rotations": `U`, `D`, `R`, `L`, `F`, `B` to any rotations, represented by length-6 arrays.
- `%4` is wrong, don't do it!!! (Why?)

- Use `struct` to implement!
- Expected Score: 100 (Cumulative: 100)

Full Solution

Alternative solution:

- Observe that arbitrary placement of dice can be achieved by 4 spins.
- Therefore, for each sequence of spins we could reduce it into a sequence with length less than 4.
- Create a `reduce()` function that you could call for every time a bracket is closed.

`string reduce(string X):`

1. Find the final placement of dice:`{ 1 2 3 4 5 6 }` after `X` operates on it.
2. For each possible combination of sequence of length 4 using "`NFBRLUD`"(e.g. `FBDR`), where `N` represent no spin, check if the final placement after the sequence have same result of the one operated by `X`, if yes then return the sequence.

Can’t Wake Syndrome

TC2322

Statistics

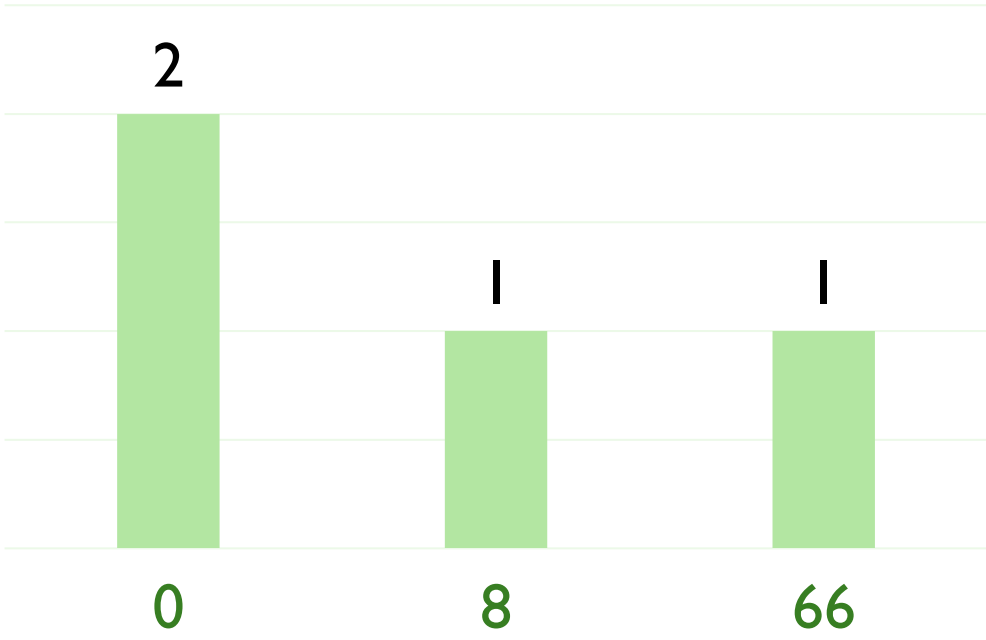
Total Score	100

Attempts	4

Subtask Performances

Subtask	Score	Solves
1	5	1
2	12	1
3	8	2
4	11	1
5	30	1
6	34	0

Score Distribution



Max Score	Jack	66

Problem Statement

- Alice can't wake up on her own. Bob calls her.
- But Bob need to find the optimal way to call Alice
- To make Alice feel happy.

- Hypothetical situation!

- Your output would be accepted if your answer has an absolute error or relative error of less than 10^{-6}: If you don't understand, just ignore it. It means that if your answer is accurate enough, you will AC! (use `long double` btw)

Subtask 1 & 2

- $N = 1 \,\&\, 2$
- Just try all the possible choices for Bob to call Alice (in case you understand the problem statement)
- Expected Score: 17 (Cumulative: 17)

Subtask 3 & 4

- Plastic subtasks(!)

- Subtask 3: $P = 100$
  - Alice always wake up when being called
  - Answer = minimum of a_i **and** $\boldsymbol{A}$
- Subtask 4: $a_i = 0$
  - If $A < 0$, not calling Alice is good, $\text{answer} = A$
  - If $A \geq 0$, repeatedly calling Alice is good, $\text{answer} = A \cdot (1 - P)^{N+1}$

- Expected Score: 19 (Cumulative: 36)

Subtask 5

- $N \leq 10$

- (Again) just try all the possible choices for Bob to call Alice
- You might use bitwise numbers to encode the states, e.g. $N = 4$:
  - `i =  0: 00000`
  - `i =  1: 00001`
  - `i =  2: 00010`
  - …
  - `i = 30: 11110`
  - `i = 31: 11111`

- Expected Score: 47 (Cumulative: 66)

Full Solution

- First note that $P \geq 50$, i.e. probability that Alice wakes up $\geq 50\%$ (Yes it is actually useful!). So the practical number of calls need ≤ 60
  - $0.5^{60} \sim 10^{-18}$, but the maximum values of a_i and A are only 10^9, so the probability that needs > 60 calls are just too small and won't affect the answer

- We use dynamic programming to solve the problem
  - `dp[n][c]`: the minimum expected angry index of Alice (except the case where Alice haven't woke up) when the last call is at `n`-th minute before Alice must wake up, and the last call is the `c`-th call
  - `dp[n][c] = min{1 <= i < n} {dp[i][c - 1] + (1-P)^(c-1) P a_n }`
  - `ans = min {dp[n][c] + (1-P)^c A }`

- Expected Score: 100 (Cumulative: 100)

Silver Legacy

TC2323

Statistics

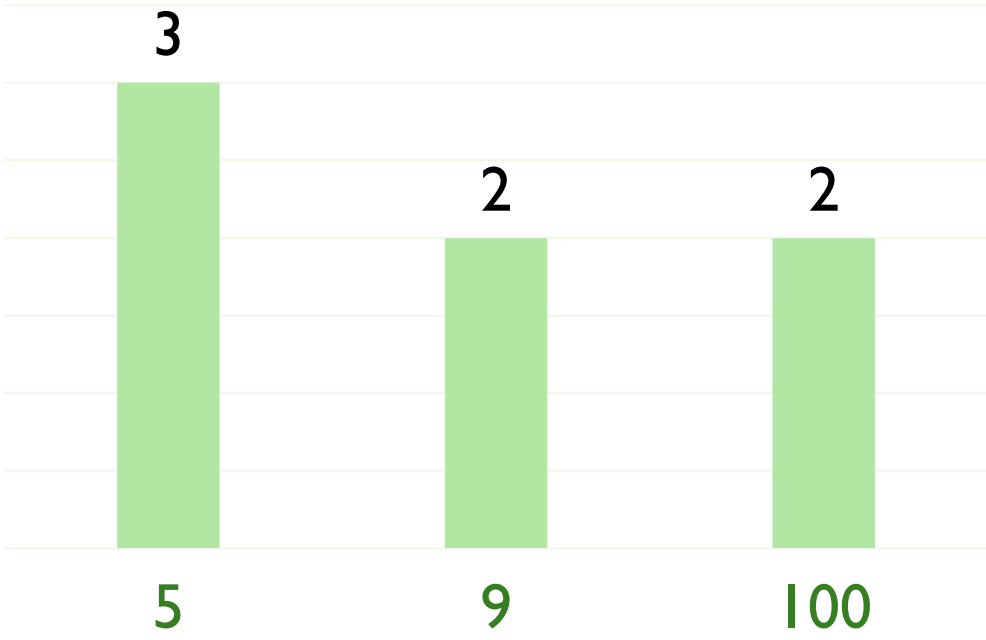
Total Score	100

Attempts	7

Subtask Performances

Subtask	Score	Solves
1	2	7
2	3	7
3	4	4
4	5	2
5	6	2
6	31	2
7	49	2

Score Distribution

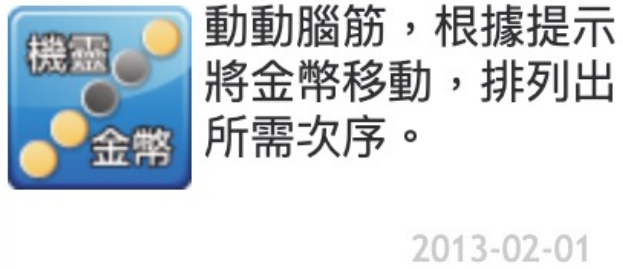


First Solve	Jack	1:22:28

Inspiration

There is an old flash game exactly like this (?) from “Small Campus” which I remember I had played.

But I couldn’t find it ☹ If anyone happens to know please tell me!



Problem Statement

- Given N silver coins on the left and N gold coins on the right
- with **one space margin at the ends**.
- Swap all the silver coins with all the gold coins with operations.

Subtask 1~5

- Hard-coded!
- This is meant for you to play.
- Is it fun?
- Hope so!

- Expected Score: 20 (Cumulative: 20)

...OK

SAMPLE TESTS

#	Input	Output
1	2	9 2 1 4 2 3 4 1 3 4 6 5 4 3 5 4 3 6 4

```
}else if (N==2){
  cout << 8 << endl;
  cout << "2 1" << endl;//101220
  cout << "4 2" << endl;//121020
  cout << "3 4" << endl;//120120
  cout << "4 6" << endl;//120021
  cout << "1 3" << endl;//021021
  cout << "3 4" << endl;//020121
  cout << "5 3" << endl;//022101
  cout << "6 5" << endl;//022110
```

```
}else if (N==2){
  cout << 8 << endl;
  cout << "2 1" << endl;
  cout << "4 2" << endl;
  cout << "3 4" << endl;
  cout << "4 6" << endl;
  cout << "1 3" << endl;
  cout << "3 4" << endl;
  cout << "5 3" << endl;
  cout << "6 5" << endl;
```

Subtask 6

- $N \leq 10$

- Any $O(N^3)$ or $O(N^4)$ would work
- Possible Idea: Write a function to swap two coins at L and $L+1$
  - Move all coins at $2, 3, \cdots, L-1$ to the left in this order
  - Jump coin $L+1$ to $L-1$
  - Move all coins at $L, L-1, \cdots, 1$ to the right in this order
- Then just repeatedly call the `swap()` function!

- Complexity: $O(N^3)$ (or $O(N^4)$)
- Expected Score: 51 (Cumulative: 51)

Full Solution

- The previous solution won't work as it is $O(N^3)$, and for $N = 100$, it requires 10^6 no. of operations but 10^5 is the limit(!)
- Just optimize it better.

- Possible Idea:
  - `SGGGGGG` → `GS GGGGG` → `G SGGGGG` → ... → `GGGGGGS` ($\sim 3n$ operations)
  - Repeat N times with some considerations

- Complexity: $O(N^2)$
- Expected Score: 100 (Cumulative: 100)

- Remember to write utility functions to guide you (e.g. range move the left, range move the right, visualize the array, etc). Much easy to debug and process, don't think this as waste of time!

Thank you!