TWGSS
OI Team

2022/23 HKOI Final Mock I

Editorial

Cut-offs

Award	TMF231 Pinball Game	TMF232 Ippyland's Got Talent	TMF233 Robin Hood and Fish	TMF234 Leaf Lover	Sum	**Overall**
Gold	76	67	58	83	284	**241**
Silver	64	56	30	67	217	**184**
Bronze	40	30	15	52	137	**116**
HM	29	17	6	44	96	**58**

Pinball Game

TMF231

Statistics

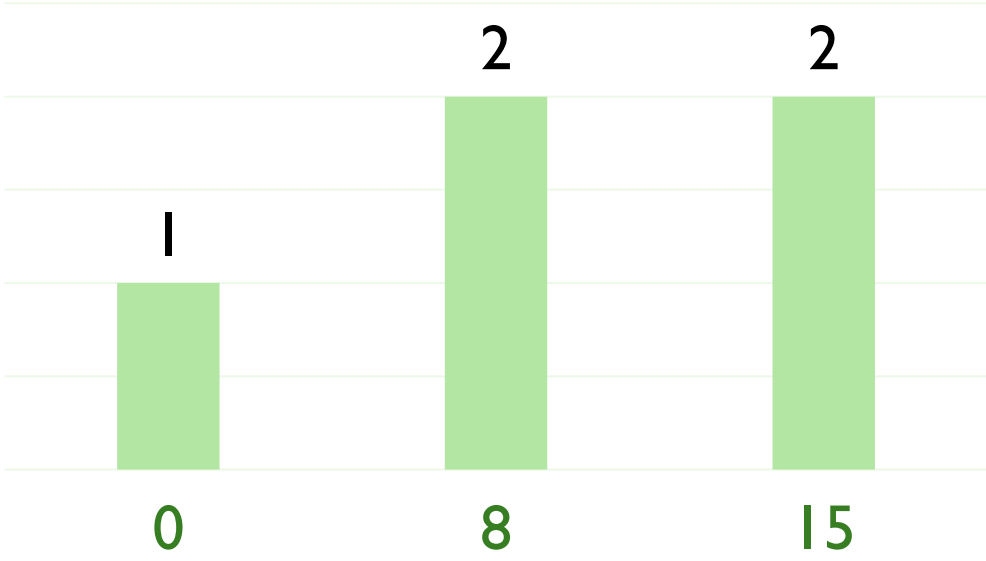
Total Score	100

Attempts	5

Subtask Performances

Subtask	Score	Solves
1	8	4
2	7	2
3	14	0
4	11	0
5	24	0
6	36	0

Score Distribution

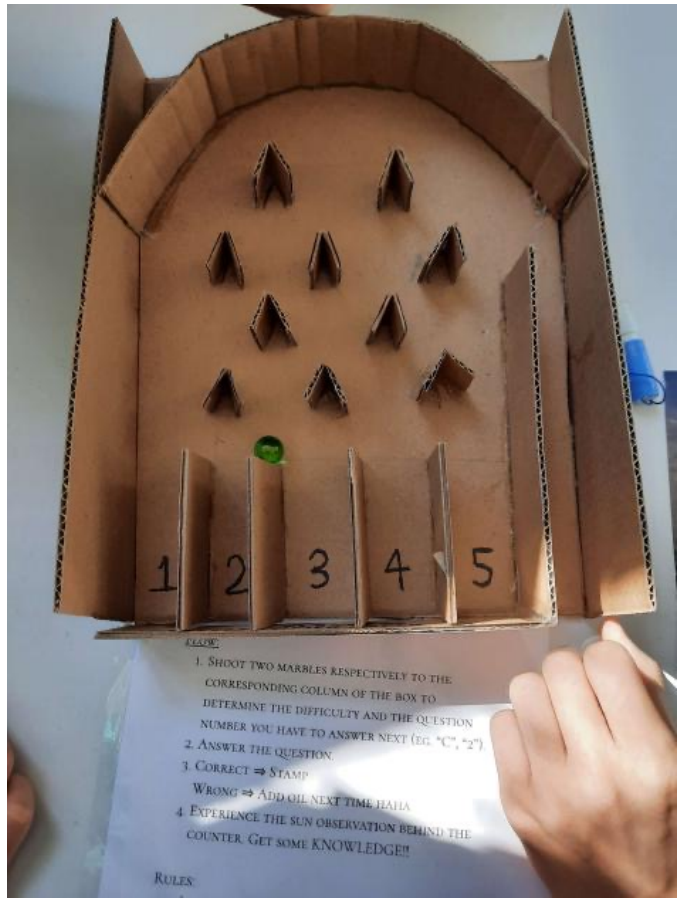


Max Score	Jack Username	15

Problem Statement

- A normal pinball machine, with '/', '\' and '.' only.
- But Felix can add extra 'bumpers',
- so as to bounce the metal balls to the correct position.

- Inspiration: I remember a long time ago, I had seen some old game machine, where if the ping pong balls are dropped in specific configuration (e.g. a line), a prize would be given.



Coincidence! And the rules are slightly different.

How to type '\'?

- HS22Q9
- \ is the escape character, so we need to represent it as '\\'

9. What is the output of the following program? 以下程序的輸出是什麼？

C++

```
int main() {
  string s = "\t\\t\\t\\t\\\t";
  cout << s.length();
}
```

A. 14
B. 4
C. 8
D. 10

Subtask 1

- $S = 0$ and $a_i = 2$ if i even, 0 if i odd for $1 \leq i \leq N$

- Understand the problem statement and the condition?
- Just output '`\.\.\. ... \.`' would work!

- Expected Score: 8 (Cumulative: 8)

Subtask 2

- $S = 0$ and $a_1 + a_2 + \cdots + a_i = i$ if $a_i \neq 0$ for $1 \leq i \leq N$
- Note that this means that for ranges of metal balls, their destination would be the rightmost one.
- Therefore, we may use '\' to repeatedly bounce the metal balls to the right. For example:

```
\..
.\.
..\
```

- Expected Score: 7 (Cumulative: 15)

Subtask 3

- $S = 0$ and $a_1 + a_2 + \cdots + a_N = N$
- The answer is always YES
- We may first split the metal balls at the top to partitions (different ranges), according to the value of a_i.
- Next, we may just use '\' and '/' repeatedly to bounce the balls to concentrate ranges of metal balls to a place.
- Note that when you implement this, you need to ensure that the routes taken by balls with different destination must be disjoint, i.e. no intersection.

- Expected Score: 21 (Cumulative: 29)

Subtask 4

- $S = 0$
- $a_1 + a_2 + \cdots + a_i$ may be smaller than N
- It is always possible as some metal balls can be trapped!
- Greedily, we may simply trap the leftmost $N - (a_1 + a_2 + \cdots + a_N)$ metal balls with '`/./ ... /./`' on the first line, then apply the idea in subtask 3 to finish the problem.

- Expected Score: 40 (Cumulative: 40)

Partial for Subtask 1-4

- All answers are YES
- So it is meaningless to give partials for the first line!

Subtask 5

- No metal balls is trapped and $a_1 + a_2 + \cdots + a_N = N$
- We first simulate the metal balls dropping through the first S rows
  - We need not deal with the case where metal balls are trapped
- Next, we may apply the variant for the algorithm solving subtask 4, which we need to find which range of columns goes to that final collector position.
  - The answer can be `NO` if after the first S rows, we cannot map each column to exactly one final collector (e.g. `1 2 0 2 0` → `2 0 3 0 0`)

- Expected Score: 64 [8] (Cumulative: 64 [48])

Full Solution

- No additional constraints
- Note $N \leq 16$.
- We may loop over all the ways to trap the metal balls in $O(2^n \cdot n)$.
  - Note we cannot trap metal balls in a single column in the middle
  - Sometimes we need 2 lines to trap a metal ball
- Then apply the solution in subtask 5.

- Expected Score: 100 [60] (Cumulative: 100 [60])

Ippyland's Got Talent

TMF232

Statistics

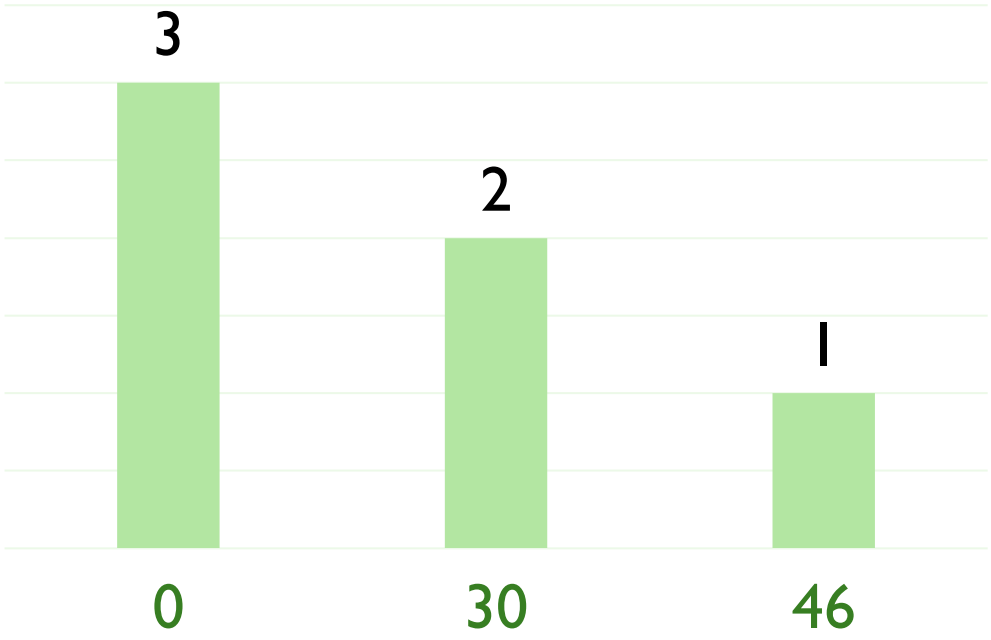
Total Score	100

Attempts	6

Subtask Performances

Subtask	Score	Solves
1	7	3
2	10	3
3	13	3
4	10	0
5	16	1
6	11	0
7	33	0

Score Distribution



Max Score	Jack	46

Problem Statement

- There are N programmers, each with ability a_i and potential b_i
- King Charles (Oops not here ☹ Busy at what leh) can increase ability of anyone by1 with a coding session
- There are at most T coding sessions

- King Charles need to select a segment of length K
- So that the final attractiveness to a specific judge is maximised
  - Alice: $a_1 + a_2 + \cdots + a_K$
  - Bob: $a_1 + 2a_2 + \cdots + 2a_{K-1} + a_K$

Subtask 1

- $K = 2$
- Note that the attractiveness to Alice and Bob are essentially the same!
- For each of $i = 1$ to $N - 1$, the maximum attractiveness is

$$\min(b_i + b_{i+1}, a_i + a_{i+1} + T).$$

- The answer is the maximum of the above over all i.

- Time Complexity: O(N)
- Expected Score: 7 (Cumulative: 7)

Subtask 2

- When the judge is Alice, only the sum of the abilities matter, i.e. adding ability to any programmer produces the same effect

- We may naively loop over all length-K ranges and add ability to any programmer as long as it does not exceed the highest potential.

- Time Complexity: $O(N^2)$
- Expected Score: 17 (Cumulative: 17)

Subtask 3

- When the judge is Alice, only the sum of the abilities matter, i.e. adding ability to any programmer produces the same effect
- ~~We may naively loop over all length-K ranges and add ability to any programmer as long as it does not exceed the highest potential.~~ Time Limit Exceeded
- Use partial sum to speed up the calculation!
- The maximum attractiveness after training for programmer $i, i+1, \cdots, i+K-1$ is

$$\min(b_i + b_{i+1} + \cdots + b_{i+K-1}, a_i + a_{i+1} + \cdots + a_{i+K-1} + T)$$

- Easy to see how to use partial sum (right?)

- Time Complexity: O(N)
- Expected Score: 30 (Cumulative: 30)

Subtask 4

- In the rest of the subtasks (70 points), the judge is Bob.
- We let the *weight* of a programmer be the multiplier that is used to multiply the ability of the programmer.
- $K = 3$
- The weights of programmers $i, i+1, i+2$ are $1, 2, 1$ respectively
- Obviously it is better to train programmer $i+1$, then i or $i+2$
- So we may just simply loop through all the ranges to obtain the answer

- Time Complexity: O(N)
- Expected Score: 10 (Cumulative: 40)

Subtask 5

- We let the *weight* of a programmer be the multiplier that is used to multiply the ability of the programmer.
- $K = N$
- The weights of programmers are $1, 2, \ldots, 2, 1$ respectively
- Obviously it is better to train programmer at the middle then on the side
- So we may just simply loop through all the programmers from greater weight to smaller weight to obtain the answer

- Time Complexity: O(N)
- Expected Score: 16 (Cumulative: 56)

Subtask 6

- When $T = 0$, you just need to calculate the sum of
$$a_i + 2a_i + \cdots + 2a_{i+K-2} + a_{i+K-1}$$, quickly.
- We can precompute (in manner of partial sum) the value of
$$ss_i = a_1 + 2a_2 + 3a_3 + \cdots + ia_i \text{ for } 1 \leq i \leq N!$$
- Then, the value to be calculated above can be 'broken down' into values of this partial sum array and the normal partial sum array ($s_i = a_1 + a_2 + a_3 + \cdots + a_i$).

- Math omitted. You can draw diagrams and write functions to help you to do the calculation correctly! (Important: reduce the number of cases to handle!!!)

- Time Complexity: O(N)
- Expected Score: 11 (Cumulative: 67)

Full Solution

- We need to combine the ideas in subtask 5 and subtask 6!
  - Subtask 5: we would greedily train the centre programmer first, then ones on the side
  - Subtask 6: use partial sum $ss_i = a_1 + 2a_2 + 3a_3 + \cdots + ia_i$ to speed up calculation
- For each range $\{i, i+1, \cdots, i+K-1\}$, we need to first binary search the *smallest weight* where all the programmers with at least that weight have reach their fullest potential. (Subtask 5)
- Then, we have the remaining sessions with the two programmers with one less that smallest weight.
- We may use partial sums to accelerate the calculations (Subtask 6)

- Time Complexity: O(N lg N)
- Expected Score: 100 (Cumulative: 100)

Robin Hood and Fish

TMF233

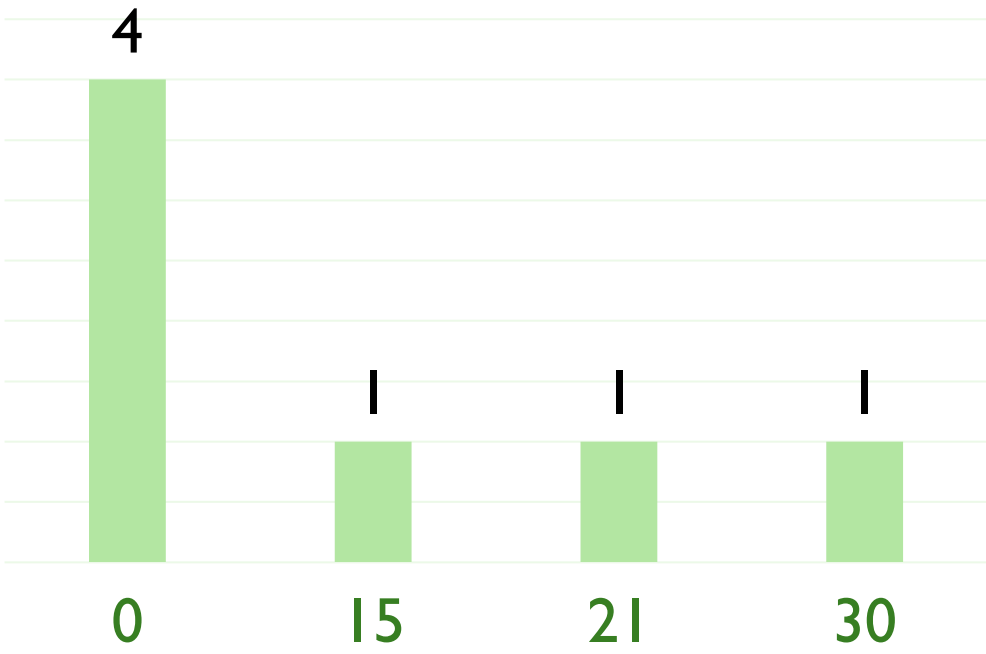
Statistics

Total Score	100	Attempts	7

Subtask Performances

Subtask	Score	Solves
1	6	3
2	9	2
3	15	2
4	28	0
5	42	0

Score Distribution



Max Score	Jack	30

Problem Statement

- There are N fishes in a row
- Fish of size S can eat adjacent fish of size T if and only if $S \geq T$
- The fish size increases from S to $S + T$, and the eaten fish disappears

Subtask 1

- $1 \leq s_i \leq 2$
- If any fish eaten any other fish, the size of it becomes at least 2 and can survive
- Therefore, just check if a fish originally has a size of at least 2 or it can eat any neighbouring fish

- Time Complexity: O(N)
- Expected Score: 6 (Cumulative: 6)

Subtask 2

- $1 \leq N \leq 5000$

- Key observation: we need only consider the case that our 'surviving' fish eat other fishes, as it is not optimal to have other fishes with larger size.
- We start with a fish, and then repeatedly find if it can eat neighbouring fish, until it is the only fish remains.
- If the process stops some time after, then it cannot survive, and vice versa.

- Time Complexity: $O(N^2)$
- Expected Score: 9 (Cumulative: 15)

Subtask 3

- $s_1 \leq s_2 \leq \cdots \leq s_N$
- Note that a larger fish always has a greater chance to survive than a smaller fish this case.
- Hence, we may simply binary search on the smallest-sized fish that will survive by solution in subtask 2.

- Time Complexity: O(N lg N)
- Expected Score: 15 (Cumulative: 30)

Subtask 4

- $s_i = 2^k$ for some non-negative integer k for $1 \leq i \leq N$
- There are only O(lg N) possible values
- Hence, you may build a partial sum where all fishes with size not exceed 2^k is accounted.
- Key Idea: Speed up the naïve process by 'jumping' with the partial sum instead of just move left or right by 1

- Time Complexity: $O(N \lg^2 N)$
- Expected Score: 58

Full Solution

- Key Idea: Speed up the naïve process by 'jumping' with the partial sum instead of just move left or right by 1
- We may pre-compute the maximum values and sum of each range of size 2^k so that we may move the bounds efficiently

- ***When a fish cannot eat a fish outside the range, this means that fish has a larger size. If we later come back to eat that fish, the size of the fish must be at least doubled!
- Therefore, the optimal route for the fish must be eat to the left, to the right, repeat for not more than $\lg \max\{s_i\}$ times.

- Time Complexity: O(N lg N lg max{s_i})
- Expected Score: 100 (Cumulative: 100)

Leaf Lover

TMF234

Statistics

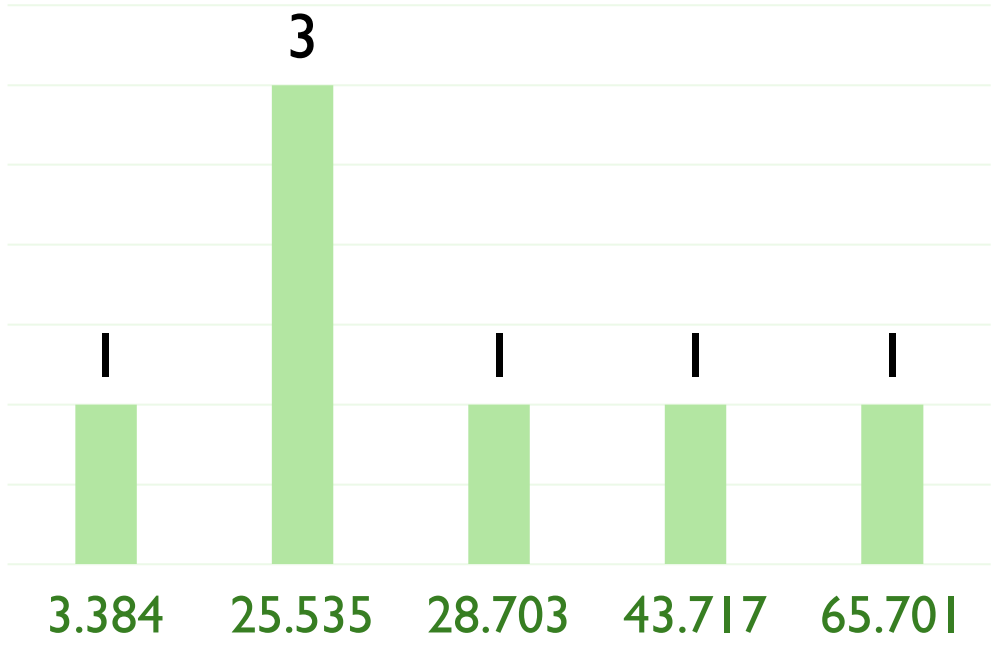
Total Score	100

Attempts	7

Subtask Overview

Subtask	Score	Scoring
1	9	Partial Score
2	24	
3	13	
4	21	
5	33	

Score Distribution



Max Score	Username	65.701

Problem Statement

- Given a grid of $N \times M$
- Maximise the number of leaves
  - $\text{Degree} = 1$ (1 adjacent tree cell)
- while the tree must be connected

```
###.
#.#.
.#.#
.###
```

Failed Example

Heart-shaped Leaf!

Subtask 1

- $N = M = 9$
- Simply hard-code!
- Author's best: 26 (No full solution for this problem)

```
.#.#.#.#.
#########
.#.#.#.#.
##..#.#.#
.########
##..#.#.#
.#.#.#.#.
#########
.#.#.#.#.
```

- Full Score: 9

Subtask 2

- $3 \leq N, M \leq 6$
- Hard-code / test small cases!

- Full Score: 24

Subtask 3

- Look for patterns!

.#.#.#.

#######

.#.#.#.

$N = 3, M = 7, \#\ \text{leaves} = 8$

- Full Score: 13

Subtask 4

- Also look for patterns!

```
.#.#.#.
#######
.#.#.#.
###.###
```

$N = 4, M = 7, \#\ \text{leaves} = 10$

- Full Score: 21

(Almost) Full Solution

- You may extend the solution for $N = 3$. However, you cannot simply merge them as the neighbour cells is going to merge together and reduce the number of leaves.

- You need to make it 'interspersed' so that the number of leaves can be maximised!

```
.#.#.#.#.
#########
.#.#.#.#.
##..#.#.#
.########
##..#.#.#
.#.#.#.#.
#########
.#.#.#.#.
```

$N = M = 9, \#\text{ leaves} = 26$

Thank You!