TWGSS OI Team

2024/25 Team Formation Test

Editorial

Nip Kit Fung Lo Ting Wai

2 October 2024

Standard Charter

TF251

Problem Statement

There are N tiles falling down in four lanes which form a pattern.

Determine whether the pattern is “standard”, that is while tapping all of them with alternating hands, two hands do not cross each other.

Fun fact: this problem was inspired by a rhythm game called *Arcaea* because of its infamous difficulty including crossing hands.

Another fun fact: this problem has totally no connection with the Standard Chartered Bank.

Subtask 1

$N = 2$

There are only TWO tiles in the pattern!

You can compare the value L_1 and L_2...

Or alternatively write an if-statement for all $4^2 = 16$ combinations if you have the time!

Expected Score	5	Cumulative	5

Subtask 2

$N = 3$

There are only THREE tiles in the pattern!

You can also compare the three values with different cases, but of course you can also code all $4^3 = 64$ cases if you think you are smart!

Expected Score	9	Cumulative	14

Subtask 3

$1 \leq L_i \leq 2$

The tiles only fall from lane 1 and 2!

There are only two situations where the pattern is not “standard”:

- when you tap lane 1 with right hand and lane 2 with left hand
- when you tap lane 2 with left hand and lane 1 with right hand

Apply this rule while looping through the sequence and break out of the loop when one of the two cases happens.

Expected Score	**11**	Cumulative	25

Full Solution

Apply the same rule as Subtask 3, except we compare the two values instead of 1s and 2s only.

Remember that the starter hand may be different for patterns starting at lanes 1, 2 and lanes 3, 4.

Expected Score	**40**	Cumulative	40

Tam’s Horror

TF252

Problem Statement

Given a list of noodles, soup, toppings and a range of spicinesses and the choice for each of them, calculate the price of an order.

This type of problem is already very familiar, right?

I sincerely hope it is :)

Subtask 1

$B =$ `Rice Noodles`

$S =$ `Clear Soup` or `Suan La Soup`

$N = 1$

$T_i =$ `Mushroom`, `Fishball`, `Pork` or `Beef`

Oh no! So many text in the first subtask!!

But if you think about it well enough, you will realise only the types of soup and toppings matter, as the other aspects are fixed.

As there are only one topping and the toppings are one word only, you do not have to worry about parsing the strings for now. You can check the first character of S and T_i only and add accordingly.

Expected Score	**7**	Cumulative	7

Subtask 2

$L = 100\%$

$N = 1$

$T_i =$ `Mushroom`, `Fishball`, `Pork` or `Beef`

All types of noodles and soups are possible to appear in the test cases.

You can also check the first characters of B and S, but you may notice that three types of soup start with `S`! What should we do?

Just check the second character if the first character is `S`! Another subtask done!

Expected Score	9	Cumulative	16

Subtask 3

$N = 1$

$T_i =$ `Mushroom`, `Fishball`, `Pork` or `Beef`

The spiciness can vary now!

You can actually input L as an integer variable and the `%` symbol will be omitted automatically!

But of course anything works as long as the value is correct.

The cost for spiciness could be calculated by $L \times \frac{2}{5}$ for your reference.

Expected Score	**27**	Cumulative	27

Full Solution

Now that every aspect of the menu is done, except toppings.

Here is one of the ways to parse the topping strings:

- Use `while(cin >> s)` to input all the strings and push them back into an array. (possible since there are no more input next)
- Iterate through the array. If the string matches the first few words of a topping, add up the cost and skip to the position of the next topping.

Remember to check for commas attached at the end of the strings sometimes!

Of course, any method works as long as the calculation is correct.

Expected Score	**60**	Cumulative	60

Definicraft Time

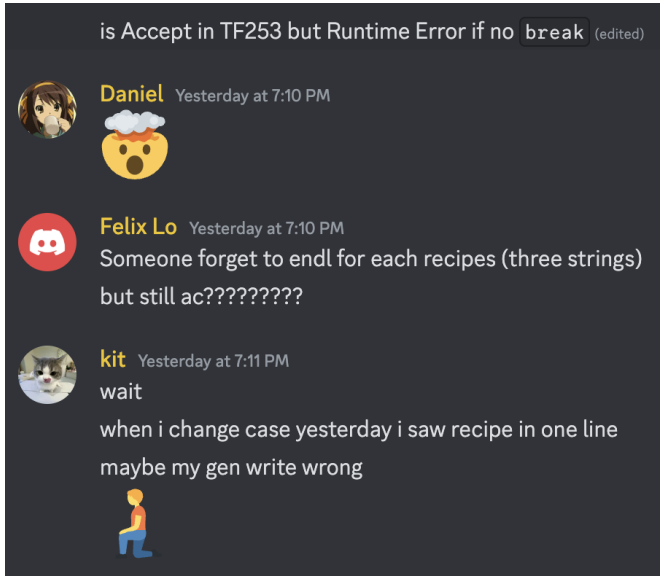
TF253

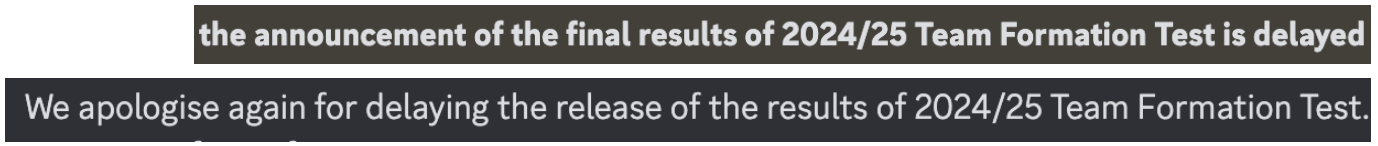
Before We Start...

The problem setter (me) sincerely apologises again for the weak test cases 🙇 , which caused the postponement of the release of the TFT results for two days.

The problem setter also appreciates the perseverance of the judge server for rejudging this memory-heavy problem for three whole times and not breaking down.

Anyways, let's talk about the problem!





Problem Statement

An operation called “crafting” involves crafting two items A and B into one.

- If there exists a recipe for crafting A and B together, the new item name will be the one specified in the recipe.
- Otherwise the new item name will be AB.

Given a list of items to be used one by one and create a complex item, find the resultant name of that item.

Subtask 1

$N = 2$
$M = 0$

There are only two base items and there are no recipes.

You can output K strings according to the item numbers `1`s and `2`s.

Easy 8 marks in your pocket!

Expected Score	**8**	Cumulative	8

Subtask 2

$K = 2$
$M = 0$

There are only two items to be used and no recipes again!

Similar to Subtask 1, you can just output the respective string for the item number.

Another very easy 11 marks in your pocket!

Expected Score	**11**	Cumulative	19

Subtask 3

$K = 2$

Recipes start appearing!

But since $K = 2$, you can loop through the recipe lists and see if the two items have a recipe for it.

- If so, output the new name directly.
- Otherwise, output the normal names.

Expected Score	**25**	Cumulative	33

Subtask 4

$2 \leq K \leq 100$

Similar to Subtask 3, you may also search through the whole list of recipe to look for the recipe for the top two items on the list.

However, the full solution may be less complicated in this case and this subtask may seem to be not so important.

Another 17 marks in your pocket, I guess? 

Expected Score	**42**	Cumulative	50

Full Solution

What is an effective way to store and retrieve the complicated recipes? A data structure!!

Use `map<pair<string, string>, string>` to store the recipes.

For further information on how to use map, please refer to the C++ Foundation Course slides. :)

Accepted!

... you sure?

Full Solution

Oh no! Why am I getting TLEs in subtask 1 and 5?!!?

	Points	Constraints
1	8	$N = 2$ $M = 0$
2	11	$K = 2$ $M = 0$
3	14	$K = 2$
4	17	$2 \leq K \leq 100$
5	25	No additional constraints

Subtask	Verdict	Score
1	Time Limit Exceeded	0 /8
2	**Accepted**	**11** /11
3	**Accepted**	**14** /14
4	**Accepted**	**17** /17
5	Time Limit Exceeded	0 /25

If you are using `map<pair<string, string>, string>` with `.count()` (or `.find()`), your program will be not fast enough apparently!

There are some ways to optimise this!

Full Solution Optimisation

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

map<pair<string, string>, string> conv;
string item[100009];
int temp;
string ans = "";

int32_t main() {
  //input n, m, k
  //input m recipes in both order of a&b

  cin >> temp;
  ans = items[temp];

  for(int i=2;i<=k;i++){
    cin >> temp;

    if(conv.count({ans, items[temp]})){
      ans = conv[{ans, items[temp]}];
    } else {
      ans += items[temp];
    }
  }

  cout << ans << endl;
}
```

This code is one of the possible solutions that gets TLE using std::map.

You may notice that for every item on the list, the code will always find whether there exists a recipe that describes the current item and the next item using `.count()`.

Note that the time complexity for looking up is O(lgN × length of `ans`), which means that the runtime will become slower when the string gets longer!

Full Solution Optimisation

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

map<pair<string, string>, string> conv;
string item[100009];
int temp;
string ans = "";

int32_t main() {
  //input n, m, k
  //input m recipes in both order of a&b

  cin >> temp;
  ans = items[temp];

  for(int i=2;i<=k;i++){
    cin >> temp;
    if(ans.size() <= 20 && conv.count({ans, items[temp]})){
      ans = conv[{ans, items[temp]}];
    } else {
      ans += items[temp];
    }
  }

  cout << ans << endl;
}
```

However, since $1 \leq \text{length of } S_i, A_i, B_i, C_i \leq 20$ as mentioned in the problem statement, it is guaranteed that there must be no recipe for strings that have length greater than 20.

Therefore, by editing the condition to filter out cases where the length of the current item name is already greater than 20, you can save many time looking up in your map.

Expected Score	**75**	Cumulative	75

King’s Territory

TF254

Problem Statement

Choose $1 \leq l_1 \leq r_1 < l_2 \leq r_2 < \cdots < l_K \leq r_K \leq L$
and maximise the minimum value of
$(r_i - l_i + 1) \times$ number of cities between l_i and r_i.

Notice that there should be at least 1 city in each interval.

We want the value of all districts be averaged.

Subtask 1

$N = K = 2$
$1 \leq L \leq 10^6$

There are only two cities and each city should belong to one district.

By greedy,

- $l_1 = 1$
- $r_2 = L$
- $r_1 + 1 = l_2$.

Given the correct r_1, you can calculate the answer easily.

How many possible r_1 are there? Only L !

O(L)	Expected Score	7	Cumulative	7

Subtask 2

$N = 2$

Now, L can be as large as 10^9. How can we find the correct r_1?

Using the solution of Subtask 1,
we may notice that the answer is usually $\frac{L}{2}$.

When will it be smaller? When $P_2 < \frac{L}{2}$ or $P_1 > \frac{L}{2}$. Why?

Therefore $P_1 \leq r_1 < l_2 \leq P_2$, greedy!

O(	**1**	)	Expected Score	**15**	Cumulative	15

Subtask 3

$N = 3$
$1 \leq L \leq 10^6$

Case 1 $K = 2$

If we know r_1, we can calculate the answer easily.

How many possible r_1 are there? Only L!

Case 2 $K = 3$

How many possible r_1 and r_2 are there? ~~Only L~~ L^2!

Subtask 3

$N = 3$
$1 \leq L \leq 10^6$

Now, we have two pointers r_1, r_2 to iterate. Really?

As there are three cites and three district, each district should contain exactly one city. Therefore, $P_1 \leq r_1 < l_2 \leq P_2 \leq r_2 < l_3 \leq P_3$.

When we are iterating r_1, there should be two cities and two districts left.
How to split the remaining two districts? Subtask 2!
How many possible r_1 are there? Only L!

O(**L**)	Expected Score **30**	Cumulative 30

Subtask 4

$N = K$
$1 \leq L \leq 10^6$

Each district should contain exactly one city.

Now, we have $K - 1$ pointers $r_1, r_2, \cdots, r_{K-1}$ to iterate. Really?

We only care about the minimum value among all K districts.

Let the value of the first district be the minimum value among all K districts.

Therefore, when r_1 is increased, $r_2, r_3, \cdots, r_K$ have to increase too. If a district contain two or more cities, terminate immediately. The total possible increment of $r_1, r_2, \cdots, r_K$ is bounded by L.

O(**L**)	Expected Score	**32**	Cumulative	47

Subtask 5

$1 \leq L \leq 10^6$

Each districts can contain multiple cities.

- Use Subtask 4 solution.
- Keep track of the number of cities in each districts.
- Done.

Use `long long`!

O(L)	Expected Score 54	Cumulative 69

Full Solution

Use Subtask 5 solution.

Notice that if the minimum value among all K districts can be x,
it can also be $x-1$ if $x > 1$.

Lets think an imaginary boolean array b where the i^{th} element be true if and only if the minimum value among all K districts can be i.

This array is automatically sorted.

What we want to find is the largest i where $b[i]$ is true.

Binary search!

O(**N lg(NL)**)	Expected Score	**90**	Cumulative	90

Robotic Cleaner

TF255

Problem Statement

A robot will keep moving right until it is blocked, move up, moving left until it is blocked, move up, moving right...

again and again until it cannot move up.

Calculate how long will it move.

Subtask 1

$N = 3$
$M = 2$

How many possible situations are there?

$2^5 = 32$!

Use a lot of `if` (about 9).

Free points!

O(	1	)	Expected Score	13	Cumulative	13

Subtask 2

$1 \leq N, M \leq 1000$

Simulation.
Follow the instructions.

Free points again!

O(**NM**)	Expected Score **33**	Cumulative 33

Subtask 3

$K = 1$

Notice that the robot can walk freely most the time.

It will walk at least $(r_1 - 1) \times M$ cells.

If it terminates, terminate.

If it is blocked, calculate the cells which are not cleaned.

It will walk the final $(N - r_1) \times M$ cells.

Use `long long`!

O(	**1**	)	Expected Score	**12**	Cumulative	45

Subtask 4

$r_1 = r_2 = \cdots = r_K$

All of the obstacles are located at the same row.
Similar to Subtask 3, but check which obstacle will block the way first.

Use `long long`!

O(K)	Expected Score 26	Cumulative 59

Subtask 5

$K = 2$

If two obstacles located on the same row, Subtask 4 solution.
If they are located on different rows, run Subtask 3 solution twice.
Special handle the termination because of the second obstacle.

Use `long long`!

O(	1	)	Expected Score	43	Cumulative	76

Full solution

Notice that there are at most K rows contain obstacle.

Therefore, combine Subtask 4 and Subtask 5 solution K times.

Use `long long`!

O(**K lgN**)	Expected Score **105**	Cumulative 105

Governor’s Battle

TF256

Problem Statement

What an amazing square number: **T**he **F**antastic **256**!

For every soldier i,

find the soldier l ($l < i$) and r ($i < r$) such that

- $S_l, S_r > S_i$
- $\mathrm{abs}(i - l)$ and $\mathrm{abs}(i - j)$ are minimised.

If no such a soldier l or r exists, output `-1` instead.

In the following pages, we only focus on finding l since the way of getting r is basically the same but with flipped direction.

Subtask 1

$N = 3$

Notice that the actual values of S_i aren't important.
We only care about the order of values of S_i.

How many possible orders are there?
$3^3 = 27$

Use a lot of `if`.

O(	1	)	Expected Score	10	Cumulative	10

Subtask 2

$1 \leq N \leq 8$

Notice that there are only 2^{2N} possible answers for each soldier.

We can check each answer in O(N).

Loop through each of the N soldiers.

$N \times N \times 2^{2N} = 8 \times 8 \times 2^{16} = 4194304 \approx 4.2 \times 10^6$

O($2^{2N} N^2$)	Expected Score	29	Cumulative	29

Subtask 3

$1 \leq S_i \leq 2$

- If $S_i = 2$, no soldier l fulfils $S_i < S_l$.
- If $S_i = 1$, only soldiers l where $S_l = 2$ fulfil $S_i < S_l$.

Use an array to save the nearest l such that $S_l = 2$ and $l < i$. How?

Let's say we have an array `A[]` and initialise an integer `now = -1`.

Just loop `j` from `1` to `N`,

- set `A[j] = now`
- if `S[j] == 2`, `now = j`

O(**N**)	Expected Score **21**	Cumulative 50

Subtask 4

$1 \leq N \leq 5000$

Simple brute force.

For every soldier i, loop all other soldiers to find the answer.

O(	N^2	)	Expected Score	45	Cumulative	66

Subtask 5

$1 \leq S_i \leq 10$

There are only 10 distinct S_i.

Similar to Subtask 3, for each S_i,
Use an array to save the nearest l such that $S_l > S_i$ and $l < i$.

O(**N**)	Expected Score **38**	Cumulative 83

Subtask 6

$S_i \neq S_j$ for $i \neq j$

For every soldier i, we want find the nearest soldier l such that $S_i < S_l$.

If we can delete all soldiers k where $S_k < S_i$, only the soldier next to it can be the possible answer and we can find it in O(1).

If $S_k < S_i$ for all i, we can delete it because k will never be an answer.

Therefore, we can sort the soldiers by S_i.

Find the answer of the soldier with smallest S_i and delete it.

Find the answer of the soldier with second smallest S_i and delete it.

...

Subtask 6

We only need a data structure to solve the problem.

- Deletion in O(1)
- Access next/previous element in O(1)

What is it?

Doubly linked list!

O(**N lgN**)	Expected Score **22**	Cumulative 105

Full Solution

Now, $S_i = S_j$ is possible.

Using Subtask 6 solution, for each soldier i, we can only find the nearest soldier l where $S_l \geq S_i$.

When $S_l = S_i$, notice that the nearest soldier of soldier i to the left is equal to the nearest soldier of solder l to the left.

Loop through the array from the left to update the correct answer.

O(**N lgN**)	Expected Score **25**	Cumulative 130