Problem Tutorial: “Paper Sheets”

0.1 Full Solution

There are only two positions for the cards, let’s consider each:

- If the cards are placed vertically, then the sheet can accommodate $\lfloor \frac{x}{w} \rfloor$ cards in width, and $\lfloor \frac{y}{h} \rfloor$ in height. This means that the total number of cards in this arrangement can be achieved as $\lfloor \frac{x}{w} \rfloor \cdot \lfloor \frac{y}{h} \rfloor$.
- If the cards are placed horizontally, then the sheet can accommodate $\lfloor \frac{x}{h} \rfloor$ cards in width, and $\lfloor \frac{y}{w} \rfloor$ in height. This means that the total number of cards in this arrangement can be achieved as $\lfloor \frac{x}{h} \rfloor \cdot \lfloor \frac{y}{w} \rfloor$.

The answer will be the greater of the two numbers, i.e., $\max(\lfloor \frac{x}{w} \rfloor \cdot \lfloor \frac{y}{h} \rfloor, \lfloor \frac{x}{h} \rfloor \cdot \lfloor \frac{y}{w} \rfloor)$.

Problem Tutorial: “Basketball Tournament”

0.2 Problem Statement

You need to learn how to distinguish between two situations.

0.3 Full Solution

- If the teams played a round-robin tournament, the total number of matches will be $\frac{n\cdot(n-1)}{2}$. Consequently, the total number of wins will be $\frac{n\cdot(n-1)}{2}$.
- If the teams played an Olympic system, the total number of matches will be $n - 1$. Consequently, the total number of wins will be $n - 1$.

Calculate the total number of wins. If it is equal to $\frac{n\cdot(n-1)}{2}$, the answer will be **Round-robin**; otherwise, it will be **Olympic**.

Problem Tutorial: “Elevator”

This problem is related to the modeling and implementation of the operation of elevators.

It is necessary to model the operation of each elevator. Let’s consider the main points in the implementation:

1. Suppose the elevator is filled with people who need floors with numbers $a_1, \ldots, a_c$, and Timur is not in the elevator. Then the time of transporting people is the sum of the time spent on stops at each floor, the trip to the maximum floor, and the descent to the first floor.

   a) The total number of different stops is equal to the number of unique numbers among $a_1, \ldots, a_c$, denote this as cnt_diff. Then the time spent on stops at the floors is $h \cdot cnt\_diff$.
   b) The time spent on the trip to the maximum floor is $lift\_time \cdot (max(a_1, \ldots, a_c) - 1)$.
   c) The time spent on the descent to the first floor is also $lift\_time \cdot (max(a_1, \ldots, a_c) - 1)$.

   In the end, the time of transporting people is $h \cdot cnt\_diff + 2 \cdot lift\_time \cdot (max(a_1, \ldots, a_c) - 1)$.

2. Suppose the elevator is filled with people who need floors with numbers $a_1, \ldots, a_t, n$, and Timur is in the elevator (who needs the floor with number n).

a) If the elevator visits floor n, then the time to travel to this floor will be equal to the sum of the time spent on stops at the floors before n and the time to travel to n. The number of stops before floor n is determined as the number of unique numbers among $a_1, \ldots, a_t$ less than n, denote this number as q. The time spent on the trip to floor n is $(n-1) \cdot lift\_time$. Thus, the time of transporting people is $(q+1) \cdot h + (n-1) \cdot lift\_time$ (added 1 to q as Timur exits at floor n).

b) If the elevator does not visit floor n, then we will run the algorithm from point **a)** for floors $n-1$ and $n+1$ respectively. In the first case, add $timur\_up\_time$ to the answer, in the second case, add $timur\_down\_time$ to the answer.

It is also necessary to consider the case when Timur walks up to floor n.

Problem Tutorial: "Triangles"

To solve the subtask where $n \leq 200$, it is sufficient to iterate through all triples of numbers $1 \leq a < b < c \leq n$, and for each triple, check that $a + b > c$. Such a solution earns 10 points.

To speed up the solution, notice that for fixed $1 \leq a < b \leq n$, all c in the range $[b+1, \min(n, a+b-1)]$ are suitable. Therefore, it is sufficient to iterate through a, b and for each pair, add the number of suitable c to the answer, which is $max(0, \min(n, a+b-1) - b)$.

For further optimization, notice that for a fixed a, there are several interesting ranges of b. The first corresponds to the solution of the inequality $a + b - 1 \leq n \Leftrightarrow b \leq n - a + 1$. The second is $a + b - 1 > n \Leftrightarrow b > n - a + 1$. Remember that $a + 1 \leq b \leq n$, so the ranges actually look like this: $b \in [a+1, n-a+1]$ and $b \in [max(a+1, n-a+2), n]$. For each b in the first range, it is necessary to add $a-1$ to the answer. And for each b in the second range, add $n-b$ to the answer.

Now notice that for $a \leq \frac{n}{2}$, the first range remains $b \in [a+1, n-a+1]$, and the second is $[n-a+2, n]$. Let's consider this case first. Then, for a fixed a, it is necessary to add $(n-2\cdot a+1)\cdot(a-1)+\sum_{n-a+2 \leq b \leq n}[n-b]$ to the answer. Denote $f(i,j)$ as the sum of natural numbers from i to j inclusive. Then, for each $1 \leq a \leq \frac{n}{2}$, it is necessary to add $(2\cdot n - 2\cdot a + 1)\cdot(a-1) - f(n-a+2, n)$ to the answer.

For the case when $a > \frac{n}{2}$, the first range degenerates into an empty one, and in all other cases, it is handled similarly.

Now we have an expression that depends only on a. It is easy to see that when expanding the brackets, the obtained value can be expressed through f and through the sum $\sum_{1 \leq i \leq k} i^2 = \frac{k\cdot(k+1)\cdot(2\cdot k+1)}{6}$. As a result, the following formula is obtained: $\frac{n\cdot(n+2)\cdot(2\cdot n-5)}{24}$. To calculate the value modulo 10^9+7 in languages without built-in support for long arithmetic, extensions of some compilers can be used, for example, $\_\_int128\_t$ in $g++$.