ABC 151 解説

writers: beet, DEGwer, kort0n, kyopro_friends, satashun, tozangezan

2020 年 1 月 12 日

For International Readers: English editorial starts on page 7.

A: Next Alphabet

一つの方法として考えられるのは、25 種類の入力に対して if 文を書き、出力する文字を決めるというものです。

しかし、英小文字は文字コード上で順に並んでいるため、入力に 1 を足すことで正解を得られます。(実は厳密にはどのような環境でも文字コード上で順に並んでいることが保証されているわけではありません。しかし AtCoder を含む現代のパソコン上では、ほとんどの場合このように順に並んでいます。)

Listing 1　C++ での実装例

```
#include <stdio.h>

char in[2];
int main(){
        scanf("%s",in);
        in[0]++;
        printf("%s\n",in);
}
```

B:Achieve the goal

$O(NK)$ 解法

最後のテストの得点を 0 点から K 点まで順に全探索し、初めて平均点が M 点以上になった時の点数が答えです。

$O(N)$ 解法

「N 科目の平均点が M 点以上」というのは、「N 科目の合計点が $N*M$ 点以上」と同じです。よって、最後のテストで は $N*M-(A_1+\ldots+A_{N-1})$ 点以上を取れば目標を達成できます。

C: Welcome to AtCoder

各問題について, 「その問題で既に AC が出たか」及び「その問題で WA が出た回数」を記録しながら, 提出を順に調べて処理します.
WA が出たときは, その問題の WA 回数を 1 増加させます
AC が出たときは, その問題での AC が初めてでなければ, 何もしません. 初めての場合は, 高橋君の正答数が 1 増え, ペナルティが「その問題で WA が出た回数」だけ増加します.
以上の処理は $O(N+M)$ であり, 十分高速です.
C++ での解答例: `https://atcoder.jp/contests/abc151/submissions/9430678`

D:Maze Master

スタート地点を決めたとき、ゴール地点はスタート地点から最も遠い箇所にすべきです。そのような地点は BFS などにより $O(HW)$ で求めることができます。したがって、各マスをスタート地点とした場合についてをそれぞれ計算することで $O((HW)^2)$ で問題が解けました。なお、ワーシャルフロイド法を用いて $O((HW)^3)$ で全点間最短距離を求めることでも解けます。

E:Max-Min Sums

A は予めソートされているとします。また、N までの階乗とその逆元を予め計算しておくことで、二項係数は高速に求められるとして良いです。

max と min について、それぞれ別々に和をとることを考えます。すなわち $\sum f(S)$ ではなく $(\sum \max S) - (\sum \min S)$ を考えます。

簡単のために、A_i は相異なるとします。$\max S$ の値としてありえるのは A の要素のいずれかです。したがって、各 i について $\max S = A_i$ となるような S がいくつあるかがわかれば、$\sum \max S$ を求めることができます。$\max S = A_i$ となる必要十分条件は「S は A_i を含み、かつ、A_i より小さなもの $K-1$ 個を含む」であるので、このようなものは二項係数を用いて直ちに計算することができます。$\sum \min S$ も同様にして求めることができます。

A_i が重複するものを持つ場合、値が等しい A_i たちの間には勝手な大小関係を入れても、上で述べた説明は成立することが確かめられます (例えば (A_i, i) の辞書順による順序を考え、$\max S = (A_i, i)$ となるものの個数を考えます)。したがって、この場合も全く同様に求められます。

F: Enclose All

この問題は以下のような問題と同値です。

問題': N 個の点 (x_i, y_i) が与えられる。それぞれの点を中心とし、半径が r の円のいずれにも含まれる点が存在するための r の最小値を求めよ。

ある半径 R に対し、考えられる N 個の半径 R の円に共通部分が存在するなら、そのどこかに 2 円の交点が存在します。この交点は、もちろん N 個の円どれからも距離 R 以下です。

よって、以下のように求める答えについての二分法をすれば答えが求められます。

- 半径 R を決める。
- N 個の半径 R の円を列挙し、2 つの円の交点を全列挙します。
- それぞれの交点に対し、N 個の点どれからも距離が R 以下かを判定します。
- 上記の条件を満たす交点が 1 つでも存在したら、求める答えは R 以下です。そうでないなら R よりも大きいです。

以上の二分法は、各ステップ $O(N^3)$ であり、50 回程度回せば十分です。(ただし誤差に気をつけてください。候補となっている点が 2 つの円の交点なので、それらの円の中心からはちょうど距離 R なので、そのような点をスキップするか、非常に小さい値を足して判定する必要があります)

半径が等しい 2 つの円の交点は、以下のように求められます。

- 2 つの交点間の距離 h を求める (三平方の定理より、円と円の距離と円の半径から求められます。ただし円と円の距離が遠く、交点が存在しない場合気をつけてください)
- 2 つの円の中心の中点から、2 つの円の中心を結ぶ直線と垂直方向に距離 h の点を取る (2 つあります)
- それら 2 つの点が 2 つの円の交点である。

A: Next Alphabet

One possible way is to write if statements for 25 kinds of input and determine the character to output.

However, the lowercase English letters are aligned consecutively on the character codes, so you can obtain the answer by adding 1 to the input. (Actually, it is not guaranteed that those are aligned consecutively on the character codes in every environment. However, in the modern computers including AtCoder, in most case they are aligned consecutively like this.)

Listing 2 Sample Code in C++

```
#include <stdio.h>

char in[2];
int main(){
        scanf("%s",in);
        in[0]++;
        printf("%s\n",in);
}
```

B:Achieve the goal

$O(NK)$ Solution

Brute force through all the possible scores from 0 points to K points, and when the average score is more than or equal to K points for the first time, the score then is the answer.

$O(N)$Solution

"The average score of K subjects are more than or equal to M" is equivalent to "the sum of scores of K subjects is more than or equal to $N * M$." Therefore, he can achieve the goal if he gets more than or equal to $N * M - (A_1 + \ldots + A_{N-1})$ points for the last test.

C: Welcome to AtCoder

For each problem, record "whether AC has already appeared for the problem or not" and "the number of WAs appeared so far for the problem," while processing the submissions in order.
If the verdict was WA, then increase the number of WAs for the problem by one.
If the verdict was AC, then if it is not the first AC for the problem, then do nothing. If it was the first one, the number of Takahashi's correct answers increases by one, and the penalties increases by "the number of WAs appeared so far for the problem."
The process above is $O(N + M)$ and fast enough.
Sample Code in C++: `https://atcoder.jp/contests/abc151/submissions/9430678`

D:Maze Master

When a starting square is fixed, it is optimal to set the goal square to be the furthest square from the starting square. Such square can be found by BFS or any other algorithms in a total of $O(HW)$ time. Therefore, by calculating for each square as a starting square, this problem could be solved in a total of $O((HW)^2$ time. Note that you can also calculate in a total of $O((HW)^3)$ time by applying Warshall - Floyd Algorithm.

E:Max-Min Sums

Assume that A is sorted beforehand. Also, you can assume that you can calculate binomial coefficients fast by precalculating the factorials till N and their inverses.

Consider calculating the sum separately for min and max. In other words, consider $(\sum \max S) - (\sum \min S)$ instead of $\sum f(S)$.

For simplicity, assume that A_i is distinct from each other. Possible value of $\max S$ is any element of A. Therefore, by counting the number of S such that $\max S = A_i$ for each i, you can find $\sum \max S$. The necessary and sufficient condition of $\max S = A_i$ is "S contains A_i, and also contains $K-1$ elements less than A_i," so such number can be directly calculated by using binomial coefficients. You can calculate $\sum \min S$ similarly.

If A_i contains duplicates, you can prove that the explanation above also holds if you assume arbitrary order between A_is with same value (for example, consider a lexicographical order of $(A_i, i$ and count the number of elements satisfying $\max S = (A_i, i)$. Therefore, you can also process in the same way in this case.

F: Enclose All

This problem is equivalent to the following problem.

Problem': You are given N points, (x_i, y_i). Find minimum r such that there exists a point contained in all the circles of radius r whose centers are the given points.

For a radius r, if there exists an intersection area of N circles of radius R, then it contains an intersection point of some two circles somewhere. Of course, the distance between this intersection point and any center of circle is less than or equal to R.

Therefore, the answer can be found by a binary search for the desired answer.

- Fix a radius R.
- Enumerate the N circles of radius R, and then enumerate all the intersection points of all pair of circles.
- For each intersection, check if the distance from the N points are less than or equal to R or not.
- If any intersection satisfies the condition above, then the answer is less than or equal to R. Otherwise it is more than R.

The binary search above costs $O(N^3)$ time for each step, and it is enough to loop about 50 times. (Be careful of the precision error. The candidate point is an intersection of two circles, so the distance from the centers of those two circles are exactly R, so you have to skip such points or judge by adding very small value.)

The intersection of two circles of same radius can be found by the following steps:

- Find the distance h between the two intersection (by the Pythagorean theorem, this can be found from the distance between the two circles and the radius of the circles. Note that you have to take care of the cases where the circles are far from each other and intersections do not exist)
- Find a vector of length h starting from the midpoint of the centers of two circles and take its endpoint (there are two of them)
- Those two points are the intersections of the two circles.