ABC 143 解説

gazelle, kort0n, beet

2019 年 10 月 19 日

For International Readers: English editorial starts on page 7.

A: Curtain

答えは，$A > 2B$ のときは $A - 2B$ であり，$A \leq 2B$ のときは 0 です.
A, B を入力として受け取り，答えを計算し，それを出力することにより，AC となります.
尚，答えを計算する際には if 文を用いても構いませんが，max 関数を用いるとより簡潔に書くことが出来ます.
以下は C++ での実装例です.

```
#include <iostream>
using namespace std;
int main(){
    int A, B, ans;
    cin >> A >> B;
    ans = max(0, A - 2 * B);
    cout << ans << endl;
    return 0;
}
```

B: TAKOYAKI FESTIVAL 2019

N が大きくないので、素直な全探索解で正答できます。

つまり、食べるたこ焼き 2 個の選び方を全探索して、それぞれの体力回復量の和をとればいいです。

この問題のように「順番を考慮しない 2 要素の選び方」を全探索する際のポイントとして、下記コードにあるように、内側のループ変数の始点を外側のループ変数 $+1$ から始めるといいです。こうすることで、内側のループで選ぶインデックスが必ず外側のループで選ぶインデックスより大きくなり、同じ選び方を 2 回見てしまうことを回避できます。

以上の全探索解の計算量は $O(N^2)$ ですが、この問題は工夫すると $O(N)$ で解くこともできます。考えてみてください。

Listing 1　C++ による実装例

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

int main() {
    int n;
    cin >> n;
    vector<int> d(n);
    for(int i = 0; i < n; i++) cin >> d[i];
    int ans = 0;
    for(int i = 0; i < n; i++) {
        for(int j = i + 1; j < n; j++) {
            ans += d[i] * d[j];
        }
    }
    cout << ans << endl;
    return 0;
}
```

C: Slimes

解法 1

同じ文字が連続する S の極大な連続部分文字列を，それぞれグループと見なします.

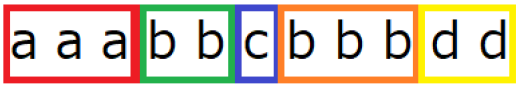

図 1　$S = aaabbcbbbdd$ に対するグループ分けの例

このとき，S の各文字間の境界であって，その両隣の文字が異なるものと，前述したグループの境界は，一対一に対応します.
以上より，前述した S 中の境界に着目することで，解を出力することが出来ます.
C++ による解答例:https://atcoder.jp/contests/abc143/submissions/8034709

解法 2

一部の言語では，文字列中の隣り合う同じ要素を、1 つのみ残して削除する関数があります. 例えば，C++ では以下のようなコードを書けば良いです.
C++ による解答例:https://atcoder.jp/contests/abc143/submissions/8034663

D: Triangles

全ての棒の組について三角形が作れるかをチェックすると，時間計算量が $O\left(N^3\right)$ となり，間に合いません.
三角形を構成する棒のうち，1 番目と 2 番目に長いものを固定します (ただし，同じ長さを持つ棒が複数存在する場合は，予め適当に大小関係を定めておきます).
このとき，3 番目に長い棒として使える棒は，「2 番目の棒より短く，一定以上の長さを持つもの」です.
このような棒の数は，予め棒の長さをソートしておくと，二分探索で効率的に求めることが出来ます.
以上より，1 番目に長い棒と 2 番目に長い棒の選び方について探索すると，時間計算量 $O\left(N^2 \log N\right)$ で答えを求めることが出来ます.
C++ による解答例:https://atcoder.jp/contests/abc143/submissions/8034764

E: Travel by Car

ワーシャルフロイド法というアルゴリズムを用いると、$O\left(N^3\right)$ で全点間の最短距離を求めることが出来ます。
まず、各町を頂点とし、各道を辺とした重み付き無向グラフを作ります。各辺の距離は、問題における道の距離とします。
このグラフにワーシャルフロイド法を適用することで、各町間の最短距離を求めることが出来ます。
次に、もう 1 つグラフを作ります。各町を頂点とし、最短距離が L 以下である町の組の間に、距離 1 の辺を張ります。
このグラフにワーシャルフロイド法を適用することで、各町間を移動する際に必要な燃料補給回数の最小値が求められます。
あとは、この結果に基づいて各クエリに答えれば良いです。時間計算量は $O\left(N^3+Q\right)$ です。
C++ による解答例:https://atcoder.jp/contests/abc143/submissions/8034843

F: Distinct Numbers

与えられた数列から、長さ $K(1 \leq K \leq N)$ の狭義単調増加数列を最大何個取り出すことができるかという問題です（要素の順番を入れ替えてもよい)。

$X(1 \leq X \leq N)$ 個の列を取り出すときの最大の長さを表す関数 $f(X)$ を考えます。

これは、 C_j を $A_i = j$ なる i の個数として、

$$f(X) = \left\lfloor \frac{\sum_{j=1}^{N} \min(C_j, X)}{X} \right\rfloor$$

という式で表せます。また、D_k を $C_j = k$ なる j の個数とすると、

$$f(X) = \left\lfloor \frac{\sum_{j=1}^{N} \min(C_j, X)}{X} \right\rfloor = \left\lfloor \frac{\sum_{k=0}^{N} \min(k, X) \times D_k}{X} \right\rfloor = \left\lfloor \frac{\sum_{k=0}^{X} kD_k + X \sum_{k=X+1}^{N} D_k}{X} \right\rfloor$$

と変形できます。したがって、$\sum_{k=0}^{X} kD_k$ と $\sum_{k=X+1}^{N} D_k$ を累積和を用いて計算することで、全ての $f(X)$ の値を $O(N)$ で求めることができます。その後、各 K について、 $K \leq f(X)$ を満たすような最大の X を求めればいいです（存在しなければ 0)。

A: Curtain

The answer is $A - 2B$ if $A > 2B$, and 0 if $A \leq 2B$.

Receive A, B as input, compute the answer, and output it, and you can get AC.

Here, when calculating the answer you may use if statement, but with max function it will be more simple.

The following is an implementation example in C++.

```cpp
#include <iostream>
using namespace std;
int main(){
    int A, B, ans;
    cin >> A >> B;
    ans = max(0, A - 2 * B);
    cout << ans << endl;
    return 0;
}
```

B: TAKOYAKI FESTIVAL 2019

Since N is not so large, you can got acccepting by simple brute force.

That is, you can perform a brute-force searching such that iterating all the possible pair of takoyaki and sum up the health points restored by them.

In order to iterate through "all the pair chosen from element," it is effectual to let the starting index of the inner loop be the current index of outer loop added by $+1$, just like the listing shown below. This way, the index of the inner loop is always strictly larger than that of outer loop, and you can avoid choosing the same pair twice.

The solution mentioned above has $O(N^2)$ complexity, but with more effort you can also solve this problem in a total of $O(N)$ time. Can you achieve it?

Listing 2 Implementation example in C++

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

int main() {
    int n;
    cin >> n;
    vector<int> d(n);
    for(int i = 0; i < n; i++) cin >> d[i];
    int ans = 0;
    for(int i = 0; i < n; i++) {
        for(int j = i + 1; j < n; j++) {
            ans += d[i] * d[j];
        }
    }
    cout << ans << endl;
    return 0;
}
```

C: Slimes

Solution 1

Let's regard each maximum consecutive subsequence of S with the same letter as a group.

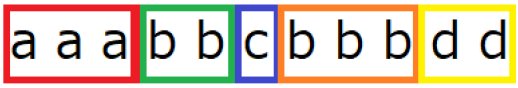

図 2　Grouping example for $S = aaabbcbbbdd$

Then the boundary between two consecutive letters of S such that its adjacent two letters are different and the boundary of the groups mentioned above corresponds one-to-one.
Therefore, by looking up the boundary in S mentioned above, the solution can be obtained.
Sample answer in C++:https://atcoder.jp/contests/abc143/submissions/8034709

Solution 2

Some languages have functions such that removes all but the first element from every consecutive group of given string. For example, in C++ you can write a code like this:
Sample answer in C++:https://atcoder.jp/contests/abc143/submissions/8034663

D: Triangles

If you loop through all the triplets of sticks and check if it is possible to make a triangle, it needs a total of $O\left(N^3\right)$ time and does not fit in time limit.
Let's fix the first- and the second- longest sticks composing a triangle (here, if some sticks have the same length, assume some order between them).
Then, the sticks that can be used as the third longest stick are those whose length is shorter than that of the second stick, but is longer than some extent.
If the sticks are sorted by their lengths beforehand, the number of such sticks can be counted efficiently by using binary search.
Therefore, by iterating through the first- and the second- longest sticks, the solution can be found in a total of $O\left(N^2 \log N\right)$ time.
Sample answer in C++:https://atcoder.jp/contests/abc143/submissions/8034764

E: Travel by Car

With Warshall – Floyd Algorithm, you can find the sortest paths between every pair of vertices in a total of $O\left(N^3\right)$ time.
First, make an undirected weighted graph such that each town corresponds to its vertex and each road corresponds to its edge.
By appying Warshall – Floyd Algorithm to this graph, you can find the shortest path between every pair of towns.
Next, make another graph; each town corresponds to its vertex, and for each pair of town such that minimum distance is less than or equal to L, there exist a vertex of distance 1.
By appying Warshall – Floyd Algorithm to this graph, you can find the minimum number of times he needs to full his tank when travelling between any two towns.
All that left is to answer each query according to the result. The time complexity is $O\left(N^3+Q\right)$.
Sample answer in C++:https://atcoder.jp/contests/abc143/submissions/8034843

F: Distinct Numbers

In this problem you are asked to find how many arrays you can take from the given array, each of which is strictly increasing array of length $K(1 \leq K \leq N)$.

Let $f(X)$ be the maximum length of array when extracting $X(1 \leq X \leq N)$ arrays.

Let C_j be the number of i such that $A_i = j$, then $f(X)$ can be represented like

$$f(X) = \left\lfloor \frac{\sum_{j=1}^{N} \min(C_j, X)}{X} \right\rfloor$$

Moreover, let D_k be the number of j such that $C_j = k$, then it can be deformed like

$$f(X) = \left\lfloor \frac{\sum_{j=1}^{N} \min(C_j, X)}{X} \right\rfloor = \left\lfloor \frac{\sum_{k=0}^{N} \min(k, X) \times D_k}{X} \right\rfloor = \left\lfloor \frac{\sum_{k=0}^{X} kD_k + X \sum_{k=X+1}^{N} D_k}{X} \right\rfloor$$

Therefore, by calculating $\sum_{k=0}^{X} kD_k$ and $\sum_{k=X+1}^{N} D_k$ with cumulative sum, you can calculate all the value of $f(X)$ in a total of $O(N)$ time. Then, for each K, you can find a maximum X such that $K \leq f(X)$ (0 if there doesn't exist).