ABC 132 解説

DEGwer, gazelle, potetisensei, tozangezan, yuma000

2019 年 6 月 29 日

A: Fifty-Fifty

以下のように、3 通りに場合分けして答えを求めればよいです。

```
#include<stdio.h>
#include<algorithm>
using namespace std;
int main()
{
        char s[10];
        scanf("%s", s);
        if(s[0] == s[1] && s[1] != s[2] && s[2] == s[3])printf("Yes\n");
        else if(s[0] == s[2] && s[2] != s[1] && s[1] == s[3])printf("Yes\n");
        else if(s[0] == s[3] && s[3] != s[1] && s[1] == s[2])printf("Yes\n");
        else printf("No\n");
}
```

また、以下のように文字列を並び替えてから判定を行っても良いです。

```
#include<stdio.h>
#include<algorithm>
using namespace std;
int main()
{
        char s[10];
        scanf("%s", s);
        sort(s, s + 4);
        printf((s[0] == s[1] && s[1] != s[2] && s[2] == s[3]) ? "Yes\n" : "No\n");
}
```

B: Ordinary Number

p_i が条件を満たすことと、$p_{i-1} < p_i < p_{i+1}$ または $p_{i-1} > p_i > p_{i+1}$ のいずれかの条件が成り立つことは同値です。よって、for ループを回してこの条件を満たす要素を数えていけばよいです。

以下は C++ での実装例です。

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    vector<int> p(n);
    for(int i = 0; i < n; i++) cin >> p[i];
    int ans = 0;
    for(int i = 1; i < n - 1; i++) {
        if((p[i - 1] < p[i]) && (p[i] < p[i + 1])) ans++;
        else if((p[i - 1] > p[i]) && (p[i] > p[i + 1])) ans++;
    }
    cout << ans << endl;
    return 0;
}
```

C. Dividing Problems(writer : yuma000)

結論から言うと、

- $N/2$ 番目に難しい問題が「ARC 用の問題」、$N/2-1$ 番目に難しい問題が「ABC 用の問題」となること
- 「ARC 用の問題」の数と「ABC 用の問題」の数が同じになること

は、同値であると言えます。よって、解法は以下のようになります。

1. 問題を難易度順に昇順でソートする。
2. $N/2$ 番目の要素から、$N/2-1$ 番目の要素を引いたものを出力する。

多くの言語にはソート用のライブラリが用意されているので、それを利用するのが良いでしょう。(C++ なら std::sort)

以下が、C++ のサンプルコードです。

```cpp
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;

int main(){
    int N;cin>>N;
    vector<int>v(N);
    for(int i=0;i<N;++i){
        cin>>v[i];
    }
    sort(v.begin(),v.end());

    int answer=v[v.size()/2]-v[v.size()/2-1];

    cout<<answer<<endl;

    return 0;
}
```

D. Blue and Red Balls

K 個の青いボールを回収するのに高橋君がちょうど i 回操作をする必要があるというのは、K 個の青ボールが赤いボールによって i 箇所に区切られているということを意味します。

そこで、次のように組み合わせを考えていくことにしましょう。

1. まず、赤いボールを $N-K$ 個一列に並べます。
2. この中で赤いボールと赤いボールの間、左端、右端の中から i 箇所を選んでそこに青いボールを K 個置くことを考えます。これらの選び方は、 ${}_{N-K+1}C_i$ 通りあります。
3. それぞれの選び方について、青いボールをそれぞれの隙間に何個割り当てていくかを考えます。それぞれに 1 個以上割り当てる必要があるので、この決め方は ${}_{K-1}C_{i-1}$ 通りあります。(*)

よって、それぞれの i について、答えは ${}_{N-K+1}C_i \times {}_{K-1}C_{i-1}$ となります。コンビネーションの計算は、${}_nC_k = \frac{n!}{k!(n-k)!}$ であることを利用し、階乗、逆元、階乗の逆元を前計算しても良いですが、今回は $N \leq 2000$ なので、パスカルの三角形を上から求めていく要領で $C[i][j] = {}_iC_j$ を DP で ($C[i][j] = C[i-1][j] + C[i-1][j-1]$) 求めてもよいです。

(*) K 個のボールを並べて、 $K-1$ 個のボールとボールの間から $i-1$ 箇所選び、そこで切り分けると i 箇所に分かれる。各部分は最小 1 個のボールを含んでおり、ボールの総数は K 個になる。

E: Hopscotch Addict

言い換えると、この問題で求めたい値は「有向グラフ $G = (V, E)$ 上の、S から T への（単純パスとは限らない）路であって、路長が 3 の倍数であるようなものの内、最短の長さ（を 3 で割ったもの）」です。

これを計算するためには、以下のようにして構成したグラフ G' 上での最短経路問題を解けば良いです:

1. G' は頂点集合として $V' = \{v_t \mid v \in V, t = 0, 1, 2\}$ を持つ。すなわち、元のグラフ G の状態数を “3 倍化” する。
2. グラフ G が u から v への辺を持つ時、G' 上に 3 辺 $u_0 \to v_1, u_1 \to v_2, u_2 \to v_0$ を張る。

こうして構成された G' 上での S_0 から T_0 への最短経路（すなわち単純パスで良い）の長さは上記の求めたい値に一致します。したがって、結局これは重みなし有向グラフ上の最短経路問題に帰着され、G' の頂点数や辺数は G の高々定数倍であるため、BFS や DFS によって $O(N + M)$ で解くことが可能です。

F: Small Products

まず、$\mathrm{DP}[i][x]$: 最後の整数が x であるような i 個の整数を並べて条件を満たすようにする場合の数 とした $O(NK)$ 状態の DP を考えます。このままでは計算量が大きいので、この DP を高速化することを考えます。

正整数 x, y が $\lfloor \frac{N}{x} \rfloor = \lfloor \frac{N}{y} \rfloor = t$ を満たすとします。このとき、$\mathrm{DP}[i+1][x]$ と $\mathrm{DP}[i+1][y]$ は同じ更新式 $\sum_{z=1}^{t} \mathrm{DP}[i][z]$ で求まるため、値は等しくなります。よって、このような x, y に対する DP の状態は同一視してよいことがわかります。

上記の規則で同一視できる状態をすべて同一視すると、DP の状態数は $O(\sqrt{N}K)$ になります。これは、$x \leq \sqrt{N}$ のときは x の値が、$x > \sqrt{N}$ のときは $\lfloor \frac{N}{x} \rfloor$ の値が $O(\sqrt{N})$ 個しかないことからわかります。適切に累積和を用いれば、この DP は $O(\sqrt{N}K)$ 時間で動くように実装できるため、この問題を解くことができました。