ABC 163 解説

gazelle, kyopro_friends, nuip, latte0119, ynymxiaolongbao, yuma000

2020 年 4 月 19 日

For International Readers: English editorial will be published in a few days.

A. Circle Pond

(円の周長) = (円の半径) * 2 * (円周率)
の式より、答えを求めることができます。以下に、今回の問題の注意点をあげておきます。

円周率の求め方

あらかじめ定数として宣言しておいたり、言語によっては標準ライブラリから求めることもできます。(今回は誤差の許容範囲が広いので、実は 3.14 まで求めておけば十分です)。

浮動小数点数の出力

これは言語によって仕様が異なるため一概には言えませんが、整数型への自動的な丸めの発生等に注意しながら実装してください。

以下が、c++ のサンプルコードです。

```
#include <iostream>
#include <math.h>
#include <iomanip>

int main(){
    int R;
    std::cin>>R;

    const double pi=acos(-1.0);

    double answer=2*R*pi;

    std::cout<<std::setprecision(20)<<std::fixed<<answer<<std::endl;
```

```

    return 0;
}
```

以下が、Python のサンプルコードです。

```
print(int(input())*6.3)
```

B:homework

全ての宿題をするためにかかる日数は $A_1+...+A_M$ です。したがって、最大で $N-(A_1+...+A_M)$ 日間遊ぶことができます。

宿題を終わらせることが不可能な場合に注意してください。

```
int a[10010];
int main(){
    int n,m;
    scanf("%d%d",&n,&m);
    for(int i=0;i<m;i++)scanf("%d",&a[i]);
    int s=0;
    for(int i=0;i<m;i++)s+=a[i];
    if(s>n){
        printf("-1\n");
    }else{
        printf("%d\n",n-s);
    }
}
```

C:management

社員番号 x の社員の直属の部下は $A_i = x$ を満たします。つまり、各数が A に何度登場するかを求めればよいです。

```
int a[200010];
int ans[200010];
int main(){
    int n;
    scanf("%d",&n);
    for(int i=2;i<=n;i++)scanf("%d",&a[i]);
    for(int i=2;i<=n;i++)ans[a[i]]++;
    for(int i=1;i<=n;i++)printf("%d\n",ans[i]);
}
```

D:Sum of Large Numbers

10^{100} は非常に巨大な数であるため、M 個の数の和はほぼ $M \times 10^{100}$ になります。"端数"の影響はわずかであることから、選んだ数の個数が異なると、和が等しくなることはありません。

よって、選んだ数の個数が $K, K+1, ..., N+1$ の各場合に答えを求め、その和を取ればよいです。

M 個の数を選ぶ場合、和としてあり得る最小値は、小さい方から M 個とった場合であり、最大値は大きい方から M 個取った場合です。実は最小値と最大値の間の値は全て作ることができます。(最小値から始めて、選ぶ数を少しずつ大きくしていくことを想像するとわかります)

あとは最大値−最小値を求めることができればよいですが、この計算において $M \times 10^{100}$ は相殺されるので、初めから 10^{100} の部分を無視して計算してよいです。和を求める公式を利用する、あらかじめ累積和を求めておく、尺取法のように和を逐次更新していく、などの方法により、この値は O(1) で求めることができます。

以上より、$O(N)$ 個の場合に対しそれぞれ $O(1)$ の時間で答えを求めることができるので、全体では $O(N)$ でこの問題が解けました。

なお、この問題は $O(1)$ で解くこともできます。

E: Active Infants

はじめ左から i 番目にいる幼児の移動先を p_i とすると、求めるものは $A_i * |i - p_i|$ を足し合わせた値の最大値になります。$|x-y| = max(x-y, y-x)$ であることから、" $A_i * |i - p_i|$ をスコアに足す" と言う代わりに " $A_i * (i - p[i])$ と $A_i * (p[i] - i)$ の好きな方を選んでスコアに足す" と言い換えても同値な問題になります。

すると、$A_i * (i - p[i])$ の方を選んだ添字 i の集合に対しては A_i の値の大きい方から p_i を $1, 2, 3, \ldots$ とし、$A_i * (p[i] - i)$ の方を選んだ添字 i の集合に対しては A_i の大きい方から p_i を $N, N-1, N-2, \ldots$ とするのが最適だということが言えます。

以上のことを踏まえれば、原始的な動的計画法を用いることで $O(N^2)$ の計算量で答えを求めることができます。具体的には、$DP[x][y]$ を活発度の高い方から $x+y$ 人を既に配置しそのうち x 人では $A_i * (i - p[i])$ を選択し y 人では $A_i * (p[i] - i)$ を選択したときのスコアの最大値と定義し、$x+y$ の昇順に計算していけば良いです。

F: path pass i

各色について、その色を一度も通らないようなパスを数え上げることができれば、元の問題も解くことができます。適当な頂点を根として DFS を行います。

頂点 v では、それぞれの子 u に対して以下を計算します。- u 以下の部分木の中で、u から色が c_v であるような頂点を一度も通らずに移動できる頂点の数

頂点 v を根とする部分木に含まれる頂点の集合を S_v とすると、上記の値は以下の式で表すことができます。

$$|S_v| - (1 + \sum_{w \in S_u, c_w = c_v, f(P_{v,w}) = 2} |S_w|)$$

ただし、$f(P_{v,w}) = |\{x | x \in P_{v,w}, c_x = c_v\}|$ とします。

各色について $\sum_{w \in S_u, c_w = c_v, f(P_{v,w}) = 2} |S_w|)$ の値を計算することを考えます。これは、頂点 v に入った時点での値 X_v を保持しておき、出るときに $X_v + |S_v|$ で更新すればよいことがわかります。

詳しくは実装例を確認してください。

計算量は $O(N)$ です。

実装例: `https://atcoder.jp/contests/abc163/submissions/12125379`