ABC 129 解説

satashun, yuma000, tozangezan, potetisensei, sheyasutaka

2019/06/09

A: Airplane

3 つの空港を順に巡る方法は 6 通りあるので、全て試しその最小値を求めればよいです。ただし、3 つのコストのうち 2 つの和の形をしていることを踏まえると、3 つの和から最大値を引くことでも答えを求めることができます。

実装例をソースコード 1 に示します。

Listing 1　実装例

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

int main() {
  int P, Q, R;
  cin >> P >> Q >> R;
  cout << P + Q + R - max({P, Q, R}) << endl;
  return 0;
}
```

B: Balance

実際に $1 \leq T < N$ を全て試してみましょう。すると、それぞれのグループの和を実際に求めることで最小値を求めることができます。愚直に和を求めると時間計算量は $\mathcal{O}(N^2)$ で、正解することができます。しかし、先頭から連続する和と、全体からその和を引いたものの差の絶対値を求めていけばよいので、和を保持しながら先頭から見ていくことで $\mathcal{O}(N)$ でも解くことができます。

実装例をソースコード 1 に示します。

Listing 1　B 実装例

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

int main() {
  int N;
  cin >> N;
  vector<int> a(N);
  int sum = 0;

  for (int i = 0; i < N; ++i) {
    cin >> a[i];
    sum += a[i];
  }

  int mini = sum;
  int prefix_sum = 0;
  for (int i = 0; i < N; ++i) {
    cin >> a[i];
    prefix_sum += a[i];
    mini = min(mini, abs(prefix_sum - (sum - prefix_sum)));
  }

  cout << mini << endl;
  return 0;
}
```

C. Typical Stairs(writer : yuma000)

まず、壊れた床がない場合 $(M = 0)$ について考察します。x 段目にたどり着くまでの移動方法の数を f_x とおきます。

$x \geq 2$ の時、x 段目にたどり着く方法としては、

- $x - 2$ 段目から一歩で二段上る。
- $x - 1$ 段目から一歩で一段上る。

の二通りがあります。つまり、$f_x = f_{x-1} + f_{x-2}$ が成り立ちます。これはフィボナッチ数の求め方の式と同等です。動的計画法 (DP) を使うことで、$O(N)$ で解くことができます。

ここまで考察できれば、壊れた床がある場合もこれを応用することで解くことができます。具体的には、f_{ai} には遷移しないように DP の遷移式を改造すれば良いです。

```
#include<bits/stdc++.h>
using namespace std;
const long long mod=1e9+7;

int main(){
    int N,M;
    cin>>N>>M;

    vector<int>oks(N+1,true);
    for(int i=0;i<M;++i){
        int a;cin>>a;
        oks[a]=false;
    }

    vector<long long int>dp(N+1);
    dp[0]=1;
    for(int now=0;now<N;++now){
        for(int next=now+1;next<=min(N,now+2);++next){
            if(oks[next]){
                dp[next]+=dp[now];
                dp[next]%=mod;
            }
        }
    }

    cout<<dp[N]<<endl;
    return 0;
}
```

D: Lamp

障害物の無いマス全てに対し、照らされるマスを for ループで数えようとすると、障害物が存在しないときに最悪計算量が $O(HW(H+W))$ となってしまい、これでは時間内に答えを求めることができません。

ただし、照らされるマスを数えるのにいくつかの補助的な前計算を活用することで、それぞれのマスに明かりを設置した時に照らされるマスの個数を定数時間で求められるようになり、こうすれば $O(HW)$ で答えを求めることができます。

まず、前計算として必要なものは、

- $L[i][j]$: i 行目 j 列目に明かりを置いた時にそこから左方向に照らされるマスの個数 (i 行目 j 列目のマスを含む)
- $R[i][j]$: i 行目 j 列目に明かりを置いた時にそこから右方向に照らされるマスの個数 (i 行目 j 列目のマスを含む)
- $D[i][j]$: i 行目 j 列目に明かりを置いた時にそこから下方向に照らされるマスの個数 (i 行目 j 列目のマスを含む)
- $U[i][j]$: i 行目 j 列目に明かりを置いた時にそこから上方向に照らされるマスの個数 (i 行目 j 列目のマスを含む)

の 4 つです。

例えば、$L[i][j]$ の要素は次のように計算できます。

- i 行目 j 列目に障害物がある時は、$L[i][j] = 0$。
- そうでないとき、$j = 1$ ならば、$L[i][j] = 1$。
- そうでないとき、$j > 1$ ならば、$L[i][j] = L[i][j-1] + 1$。

これらを 4 方向に対して計算した後、i 行目 j 列目のマスに明かりを置いた時に照らされるマスの総数は、$L[i][j] + R[i][j] + D[i][j] + U[i][j] - 3$ と求められます。(これら 4 つの値はどれも i 行目 j 列目のマスを含んでいるので、重複を消すために 3 を引いています。)

E: Sum Equals Xor

一般に、$A+B$ は A XOR B 以上になります。これらの値が一致する条件は、A, B を 2 進数表記した際、「どちらの数についても 1 であるような桁」が存在しないことです。逆にそのような桁が存在した時、$A+B$ では繰り上がりが発生し、XOR では発生しないため、$A+B$ の方が真に値が大きくなります (そしてその差を打ち消すことは出来ません)。

この考察を元に桁 DP をすることを考えます。

- $\mathrm{dp1}[k] :=$ A および B の値の上から k 桁目までを確定させ、その時点で既に $A+B$ が L 以下であることが分かっているような値組 (A, B) の個数
- $\mathrm{dp2}[k] :=$ A および B の値の上から k 桁目までを確定させ、その時点ではまだ $A+B$ が L 以下になるかどうか分かっていないような値組 (A, B) の個数

として定義します。この時、$\mathrm{dp*}[k-1]$ から $\mathrm{dp*}[k]$ を計算するには、k 桁目を 0 にするか 1 にするかの 2 通りの遷移を考慮します。k 桁目を 0 にする場合、A および B の k 桁目はどちらも 0 でなければなりません。k 桁目を 1 にする場合には、$(0,1)$ および $(1,0)$ の 2 通りの遷移が考えられます。この時、dp2 に関しては、k 桁目を確定させた際に $A+B$ が L を超えてしまわないよう、慎重に計算を行う必要があります。

以上のように、L を文字列だと見做して、$O(|L|)$ で解くことができます。

F. Takahashi's Basics in Education and Learning

問題を次のように言い換えます．

- 文字列 X と 整数 s を持つ．はじめ，$X = \text{"0"}, s = A$ である．
- 1 回の操作では，X の末尾に s が連結された後，s が B 増える．
- N 回操作した後の $X \bmod M$ はいくらか．

d 桁の要素の個数を C_d とすると，$\underbrace{99\ldots9}_{d}$ 以下の要素の個数から $\underbrace{99\ldots9}_{d-1}$ 以下の要素の個数を引くことで，C_d は容易に求まります．d 桁の要素 C_d 個を X の末尾に連結する操作は，$(X, s) \mapsto (X \times 10^d + s, s + B)$ という操作を C_d 回繰り返すことと同値です．そして，この操作は以下のように行列の積の形に落とし込めます．

$$(X, s, 1) \begin{pmatrix} 10^d & 0 & 0 \\ 1 & 1 & 0 \\ 0 & B & 1 \end{pmatrix} = (X \times 10^d + s, s + B, 1) \tag{1}$$

したがって，この 3×3 行列の C_d 乗が高速に求まればよく，繰り返し二乗法によって $O(\log C_d)$ で結果が求まります．

上の計算を $d = 1, 2, \ldots, 18$ についてこの順に行えば，最終的な $X \bmod M$ の値が高速に求まります．オーバーフローに十分注意してください．