キーエンスプログラミングコンテスト 2020 解説

wo01

2020 年 1 月 18 日

A 問題

1 回のペイント操作で黒く塗られたマスを $\max(H,W)$ 個増やすことができます。よって、求める値は $M=\max(H,W)$ として、$KM \geq N$ を満たす最小の整数 K となります。

M の値を求めるには、条件分岐を用いるか、標準ライブラリとして多くの言語で用意されている関数 max を用いればよいです。

$KM \geq N$ を満たす最小の整数 K の値は、演算 / を整数除算として、

- N が M で割り切れるとき、N/M
- そうでないとき、$N/M+1$

となるので、条件分岐を用いて求めることができます。なお、$(N+M-1)/M$ とすれば条件分岐なしでも求めることができます。

以下に C++ による実装例を示します。

```
#include<cstdio>

using namespace std;

int main(){
    int H, W, N;
    scanf("%d%d%d", &H, &W, &N);
    int m = H > W ? H : W;
    int ans = N / m;
    if(N % m != 0) ans++;
    printf("%d\n", ans);
    return 0;
}
```

B 問題

この問題の原案はキーエンス社の社員により提供されました

$S_i = X_i - L_i, T_i = X_i + L_i$ とします。ロボット i の腕が動く範囲は (S_i, T_i) となります。

ロボット i を残すかどうかは座標 T_i 以上の部分に影響を与えないことに注意すると、以下のような貪欲法により答えを求めることができます。

- ロボットを T_i が小さい順にソートし、$p_1, p_2, \ldots, p_N$ とする。
- $i = 1, 2, \ldots, N$ の順に、以下のことを行う。
  - ロボット p_i の腕がすでに残すと決めたロボットの腕と重ならないならば、ロボット p_i を残すと決める。そうでないなら、残さないと決める。

以下に C++ による実装例を示します。

```
#include<cstdio>
#include<utility>
#include<algorithm>

using namespace std;

typedef pair<int, int> P;

const int MAX_N = 100000;
const int MAX_V = 1000000000;

int N;
int X[MAX_N];
int L[MAX_N];

void input(){
    scanf("%d", &N);
    for(int i = 0; i < N; ++i){
        scanf("%d%d", X + i, L + i);
    }
}

P ps[MAX_N];

int solve(){
    for(int i = 0; i < N; ++i){
```

```
        ps[i] = P(X[i] + L[i], X[i] - L[i]);
    }
    sort(ps, ps + N);
    int cur = -MAX_V;
    int ans = 0;
    for(int i = 0; i < N; ++i){
        if(cur <= ps[i].second){
            ans++;
            cur = ps[i].first;
        }
    }
    return ans;
}

int main(){
    input();
    int ans = solve();
    printf("%d\n", ans);
    return 0;
}
```

C: Subarray Sum

以下のような数列が条件をを満たします。

- $S < 10^9$ のとき、$S, S, \ldots, S, S+1, S+1, \ldots, S+1$
- $S = 10^9$ のとき、$S, S, \ldots, S, 1, 1, \ldots, 1$

ただし S は K 個並べます。

D: Swap and Flip

カードを $p_1, p_2, \ldots, p_N$ の順に並べ替えるために必要な操作回数は、$i < j$ かつ $p_i > p_j$ を満たす (i, j) の個数に一致します。この個数を転倒数といいます。どのようにカードを並べれば表面の整数が単調増加となるか考えましょう。

カード i が何回かの操作のあと左から j 番目の位置に来たとします。このとき、上を向いている面に書かれた値は、操作手順によらず、

- $i - j$ が偶数ならば A_i
- $i - j$ が奇数ならば B_i

となります。

このことを用いると、

- $dp_{mask,i} = mask \subseteq \{1, 2, \ldots, N\}$ に含まれるカードを左から何枚かのカードとして単調増加になるように並べるときの、この部分の転倒数への寄与

という動的計画法により答えを求めることができます。

E: Bichromization

与えられるグラフを G とします。

まず、明らかに不可能なケースを考えましょう。ある頂点 v が存在して、任意の v に隣接する頂点 u に対して $D_v < D_u$ が成り立つ場合、割り当ては不可能です。なぜなら、頂点 v から色が異なる頂点への最小のコストのパスを $v = v_1, v_2, \ldots, v_k$ とするとき、

- $k > 2$ ならばパス $v_2, \ldots, v_k$ が
- $k = 2$ ならばパス v_2, v_1 が

それぞれ頂点 v_2 から色が異なる頂点へのパスとなっているからです。

上記以外の場合は以下のようにして色と重みを割り当てることができます。

まず、各頂点 v に対して、それに隣接する頂点 u のうち D_u の値が最小のもの（複数ある場合、それらのうち番号 u が最小のもの）をとって p_v とします。そして、グラフ G の各頂点 v について辺 $\{v, p_v\}$ を残したグラフ G' を考えます。このグラフは森になります。

次に、頂点の色、および G' に含まれる辺の重みの割り当てを考えます。頂点の色を、各 v について 頂点 v と頂点 p_v が異なる色になるように定めます。さらに、森に含まれる辺 $\{v, p_v\}$ に重み D_v を割り当てます。これでグラフの形が G ではなく G' であれば距離の条件が満たされるようになりました。

最後に G' に含まれない辺に重みを割り当てる必要がありますが、このような辺についてはすべて重み 10^9 を割り当てれば良いことがわかります。

以上で割り当てが得られました。

F 問題

準備中