ABC 164 解説

kyopro_friends, Kmcode, latte0119, tozangezan, ynymxiaolongbao, wo01

2020 年 4 月 26 日

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

A:Sheep and Wolves

if 文などにより条件分岐を行うことで、それぞれの場合に対応した出力を行います。

C 言語での実装例

```
int main(){
    int s,w;
    scanf("%d%d",&s,&w);
    if(w>=s)printf("unsafe");
    else printf("safe");
}
```

Python での実装例

```
s, w = map(int, input().split())
if w >= s:
    print("unsafe")
else:
    print("safe")
```

B. Battle

仮にどちらかのモンスターの体力が 0 以下になっても攻撃を続けたとき、高橋君のモンスターの体力が 0 以下になるのは青木君の $\lceil \frac{A}{D} \rceil$ 回目の攻撃のとき、青木君のモンスターの体力が 0 以下になるのは高橋君の $\lceil \frac{B}{C} \rceil$ 回目の攻撃のときです。したがって、$\lceil \frac{A}{D} \rceil$ を X、$\lceil \frac{B}{C} \rceil$ を Y として、$X \geq Y$ なら高橋君の勝ち、そうでなならば青木君の勝ちです。高橋君が先攻であるため $X = Y$ のとき高橋君の勝ちであることに注意してください。

以下が、c++ のサンプルコードです。

```
#include<iostream>
using namespace std;
int main(){
    int A,B,C,D;
    cin>>A>>B>>C>>D;
    int X=(A+D-1)/D;
    int Y=(C+B-1)/B;
    cout<<(X>=Y?"Yes":"No")<<"\n";
}
```

C: gacha

S_i がすでに手に入れたものと重複しているかどうかを毎回確認していると TLE になります。

文字列を辞書順などでソートしてから重複を確認したり、set や map,dict などのデータ構造を用いて管理することで高速に求めることができます。

D: Multiple of 2019

S の長さを n、S の左から k 番目の数字を a_k とします。

(i,j) が条件をみたすのは、$a_j + 10a_{j-1} + 100a_{j-2} + ... + 10^{j-i}a_i$ が 2019 の倍数のときです。

ところで、2019 は 2 でも 5 でも割り切れないので、mod2019 では 10 に逆元が存在します。そのため、$10n$ が 2019 の倍数であることと n が 2019 の倍数であることは同値です。(2020 ではこの性質が成り立たないので、この問題では 2019 が使われています。)

左から $k+1$ 文字目以降の数字列を整数とみなした時の値 $a_n + 10a_{n-1} + ... + 10^{n-k-1}a_{k+1}$ を T_k とします $(T_n = 0)$。 すると、(i,j) が条件をみたすのは $10^{n-j}(T_{i-1} - T_j)$ が 2019 の倍数のとき、すなわち $T_{i-1} == T_j(mod2019)$ のときです (ここで上の議論を使いました)。

最終的には、 各 $T_i mod$2019 を計算し、mod の値が同じごとに個数を計算して足せば良いことになります。$T_i = 10T_{i+1} + a_i$ より、これは DP で計算でき、mod ごとに $m(m-1)/2$ (m は $T_i mod$2019 がある値になる i の個数) を足せば良いです。

E: Two Currencies

$A_{max} = max\{A_i \mid i = 1, 2, ..., M\}$ とします。銀貨を $A_{max}(N-1)$ 枚以上所持している場合の戦略を考えてみましょう。これは明らかに、最短経路に沿って目的地に向かうのが最適です (このとき消費する銀貨の枚数は高々 $A_{max}(N-1)$ 枚なので、銀貨が足りなくなることはありません)。従って、以下の制約を追加しても、問題の答えは変わりません。

追加の制約

移動の途中で所持している銀貨の枚数が $A_{max}(N-1)$ 枚を超えた場合、$A_{max}(N-1)$ 枚になるまで銀貨を捨てる

上記の制約を追加した問題に対するアルゴリズムを考えましょう。(現在の頂点, 所持している銀貨の枚数) を状態として dijkstra 法を適用すると、$O(A_{max}NMlog(A_{max}N))$ でこの問題を解くことができます。$A_{max} \leq 50, N \leq 50, M \leq 100$ なので、十分高速です。

F: I hate Matrix Construction

まず、2 進数表現したときの各ビットの値は独立に決められるので、それぞれのビットで独立に考えます。そうすると $N \times N$ の行列に条件を満たすような 0, 1 を当てはめてる問題に落ちます。

さて、各要素について、値を確定してよい部分を先に確定させましょう。
各行及び各列について、「論理積が 1 」というのが条件だったとき、その行・列のすべての要素は 1 である必要があります。
同様に、「論理和が 0 」というのが条件だったとき、その行・列のすべての要素は必ず 0 である必要があります。
さらに、確定していない各要素について、その要素が所属している行・列の条件の組み合わせが論理和、論理積にかかわらず同じ値だったとき、その要素をその値に確定しても問題ありません。
ここまでの操作を「基本的な操作」とします。

ここで、まだ確定していない要素をすべて 0 で埋めることにすると、各行・各列について「論理和が 1 」であるという条件が満たせない場合があります。
まず、行の「論理和が 1 」という条件を満たすことを考えましょう。条件を満たすには対応する行に所属する要素 1 つを適切に選んで 1 にすればよいです。
そもそも、一連の基本的な操作で条件を満たすことができなかった場合、列は「論理積が 0 」と「論理和が 0 」の場合で構成されているときのみであることがわかります。
「論理和が 0 」の場合その列の要素はすべて 0 である必要がありますので、「論理積が 0 」の列と現在見ている行に所属する要素 1 つを適切に選んで 1 にかえる必要があります。
具体的には、「論理積が 0 」の列 1 つを選んで、その列に 0 が 1 個以上残るならば、対応する要素 1 つを 1 に変えます。

列の「論理和が 1 」という条件を満たすことは同様の操作で可能です。

一連の操作に矛盾が発生した場合、すべての条件を満たす行列は構築できないことがわかります。

実装例: `https://atcoder.jp/contests/abc164/submissions/12391952`