ABC 164 解説

kyopro_friends, Kmcode, latte0119, tozangezan, ynymxiaolongbao, wo01

2020 年 4 月 26 日

For International Readers: English editorial starts on page 7.

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

(4 月 27 日: 誤りを訂正しました。)

S の長さを n、S の左から k 番目の数字を a_k とします。

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 $T_{i-1} == T_j (mod 2019)$ のときです。

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したものが答えです。$T_i = T_{i+1} + 10^{n-i-1}a_i$ 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 $0 \leq i \leq n$ 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`

A:Sheep and Wolves

Perform conditional branch using statements like if statements, and output the string corresponding to each case.

Implementation example in C:

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

Implementation example in Python:

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

B. Battle

If they continue to attack each other even after the health of either monster becomes 0 or below, then the health of Takahashi's monster becomes 0 or less in the $\lceil \frac{A}{D} \rceil$-th Aoki's attack, and the health of Aoki's monster becomes 0 or less in the $\lceil \frac{B}{C} \rceil$-th Aoki's attack. Therefore, let X be $\lceil \frac{A}{D} \rceil$ and Y be $\lceil \frac{B}{C} \rceil$, and Takahashi will win if $X \geq Y$, and Aoki will win otherwise. Note that Takahashi will win when $X = Y$ because Takahashi takes turn first.

The following is a sample code in c++.

```
#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

Checking if S_i duplicates to one of the obtained elements so far leads to TLE.

You can calculate fast by first sorting the strings in the lexicographical order and checking for duplicates, or managing them with the data structures like set, map or dict.

D: Multiple of 2019

(Corrected on April 27.)

Let n be the length of S, and a_k be the k-th leftmost digit of S.

(i, j) satisfies the condition when $a_j + 10a_{j-1} + 100a_{j-2} + ... + 10^{j-i}a_i$ is a multiple of 2019.

By the way, 2019 is not divisible by 2 or 5, so there exists the inverse of 10 in $mod 2019$. So $10n$ is a multiple of 2019 if and only if n is a multiple of n. (2020 does not satisfy such conditions, so 2019 is used for this problem.)

Let T_k be the digits after the $k+1$-th digit, represented as an integer, that is, $a_n + 10a_{n-1} + ...+10^{n-k-1}a_{k+1}$ ($T_n = 0$). Then, (i, j) satisfies the condition when $T_{i-1} == T_j (mod 2019)$ (Here the discussion above was used).

Ultimately, it appears that it is sufficient to calculate each $T_i mod 2019$, and count the number for the elements with the same mod and sum them up. Since, $T_i = T_{i+1} + 10^{n-i-1}a_i$, this can be calculated by DP, and it is sufficient to add $m(m-1)/2$ for each mod (where m is the number of i such that $T_i mod 2019$ is equal to that remainder).

E: Two Currencies

Let $A_{max} = max\{A_i \mid i = 1, 2, ..., M\}$. Consider the strategy when you have more than $A_{max}(N-1)$ silver coins. Obviously, it is optimal to go to the destination along the shortest path (during that, at most $A_{max}(N-1)$ coins are consumed, so you never run out the silver coins). Therefore, adding the constraints below does not change the answer of the problem.

Additional constraints

If the number of silver coins exceed $A_{max}(N-1)$ during the travel, discard them until it becomes $A_{max}(N-1)$

Let us consider the algorithm to the problem under the constraints above. By applying Dijkstra method while holding the state of (current vertex, the number of silver coins), the problem can be solved in a total of $O(A_{max}NMlog(A_{max}N))$ time. Since $A_{max} \leq 50, N \leq 50, M \leq 100$, it is fast enough.

F: I hate Matrix Construction

First, when binary notated, the value of each bit can be determined independently, so we consider each bit independently. Then this problem falls into a construction of matrix of size $N \times N$ consisting of 0 and 1 that satisfies the conditions.

Now, let us determine the elements where it can be determined first.
For each row and column, if the condition is that "the logical product is 1," then all the elements of the row/column has to be 1.
Similarly, if the condition is that "the logical sum is 0," then all the elements of the row/column has to be 0.
Moreover, for each undetermined element, if the conditions of the row column that the element belongs was the same, no matter it is of logical sum or product, then the value of the element can be determined to that value.
We will call the operations so far as "basic operations."

Then, if the undetermined elements are all filled by 0, then some conditions of rows/columns that "the logical sum is 1" may not be satisfied.
First, let's consider satisfying the condition that "the logical sum of a row is 1." To satisfy the condition, we can appropriately choose one of the elements belongs to the corresponding row and set it 1.
In the first place, it can be seen that a condition could not be satisfied by the basic operations only if the columns are composed of "the logical product is 0" and "the logical sum is 0."
If "the logical sum is 0," then all the elements of the column has to be 0, so we have to choose such a appropriate element that belongs to the column of "the logical product is 0" and the column we are currently looking at, and set its value 1.
Specifically, choose a column such that "the logical product is 0," and if 0 remains in the column, then pick a corresponding element and set it 1.

Satisfying the conditions of "the logical product of the column is 1" can be satisfied by the similar operations.

If any of those operations leads to contradiction, then it appears that a matrix that satisfies all the conditions cannot be constructed.

Sample Code: `https://atcoder.jp/contests/abc164/submissions/12391952`