きたむーの独り言。

前書き

コンテスト終了時に起きていられる自信がないので、普段 Twitter に書くことをちょっとだけ書いておきます。

解説は次のページから始まります。これめっちゃ大事。別にこの 1 枚目読まなくていいです。

きたむーの彼女

これはいろはコンですが、彼女はいろはちゃんではありません。

鬼の実行時間制限と実行メモリ制限

Day2 のときに Python が落ちて叩かれてしまったのですが、僕の問題はやたら時間とメモリの制限が厳しいです。実は「あるよるのできごと」ももともと「1 秒、384MB」でした。これは、

・メモ化再帰を使って無駄な処理を省くと実は $O(N^4)$ ($N \leqq 100$)でも 50ms かからないことに気付いてほしい(C++想定解は 34ms、Python でも 1 秒あれば流石に足りると思われる)

・適切な変数型を選べるようになってほしい

というのが理由です。前者は割と汎用的なテクだと思いますし、後者も bool 型の代わりに long long 型を使うのは流石にエレガントじゃないと思うので・・・。

なお、制限変更で、多分ループで DP しても long long 型を使っても通るようになったと思うのですが、この変更がコンテスト 2 日前のことで、解説の修正が間に合わなかったので、この後の解説では bool 型を使いましょうとなっています。

writer 勢の強い人々がそうだったので僕の鬼の制限に疑問のある人はたくさんいると思いますが、僕は所詮青色なので許してください。

あと、もしかしたらこの変更で $O(N^5)$が通るようになっちゃったかもしれません。もうここまでくると問題として破綻してる気がしますが、今更差し替えもできないので仕方がないです。すみません。今後有志コンを開く方は余裕をもって作問をし、余裕をもってテストをすることを強く勧めます。

配点が適正か

適正ではないです。300 ではないと思います。絶対。でも、100 ずつ上がってるのが美しい、みたいな感じで 300 が維持されました。ごめんなさい。

大荒れのいろはコンだった気がしますが、楽しんで頂けたなら幸いです。

来年の GW は 10 日間×10 問の「ちはやふる」コンテストを僕が主催します。Twitter で writer を募集しているので是非。(大嘘)

2019/5/2　きたむー(@Pro_ktmr)

いろはちゃんコンテスト解説

Day4-A:あるよるのできごと

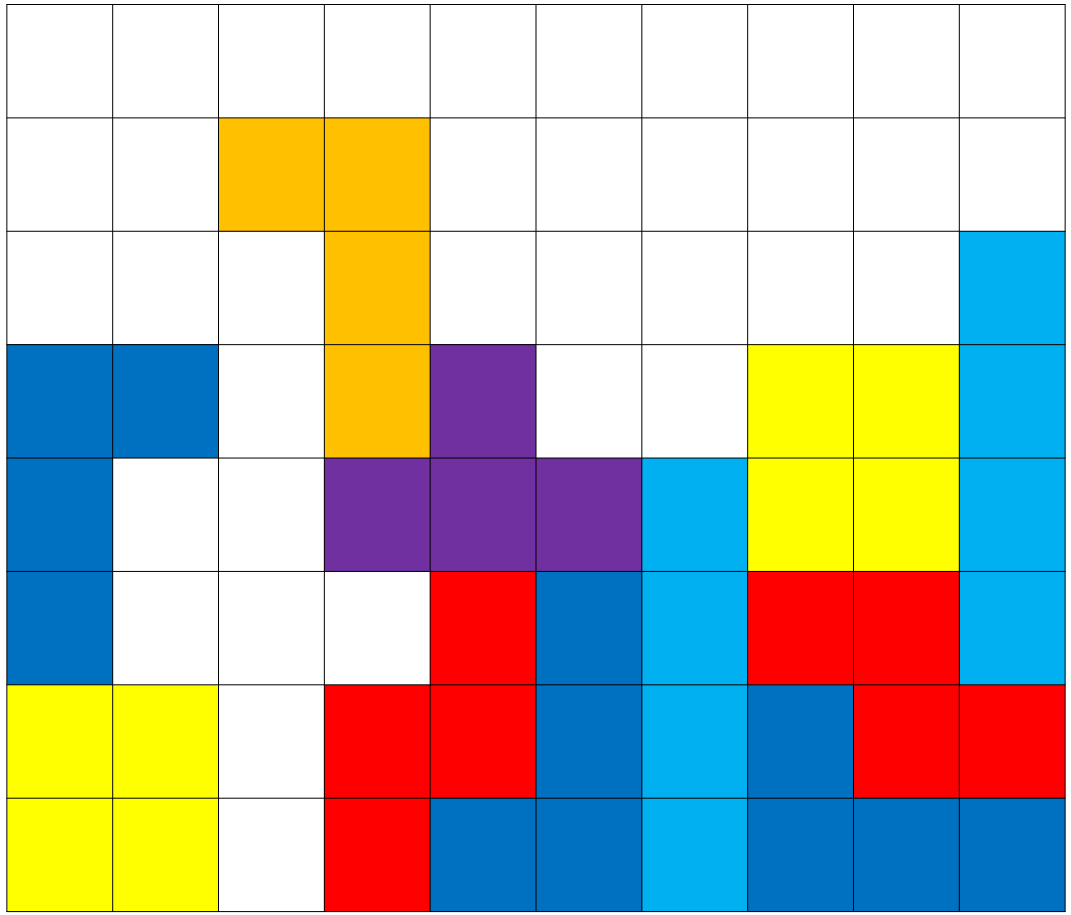
@Pro_ktmr

注意

- 内容の正確性には十分注意していますが、間違った情報が含まれることもあります。何かお気づきのことがあればご連絡ください。
- このスライドのサンプルコードはC++で実装されています。

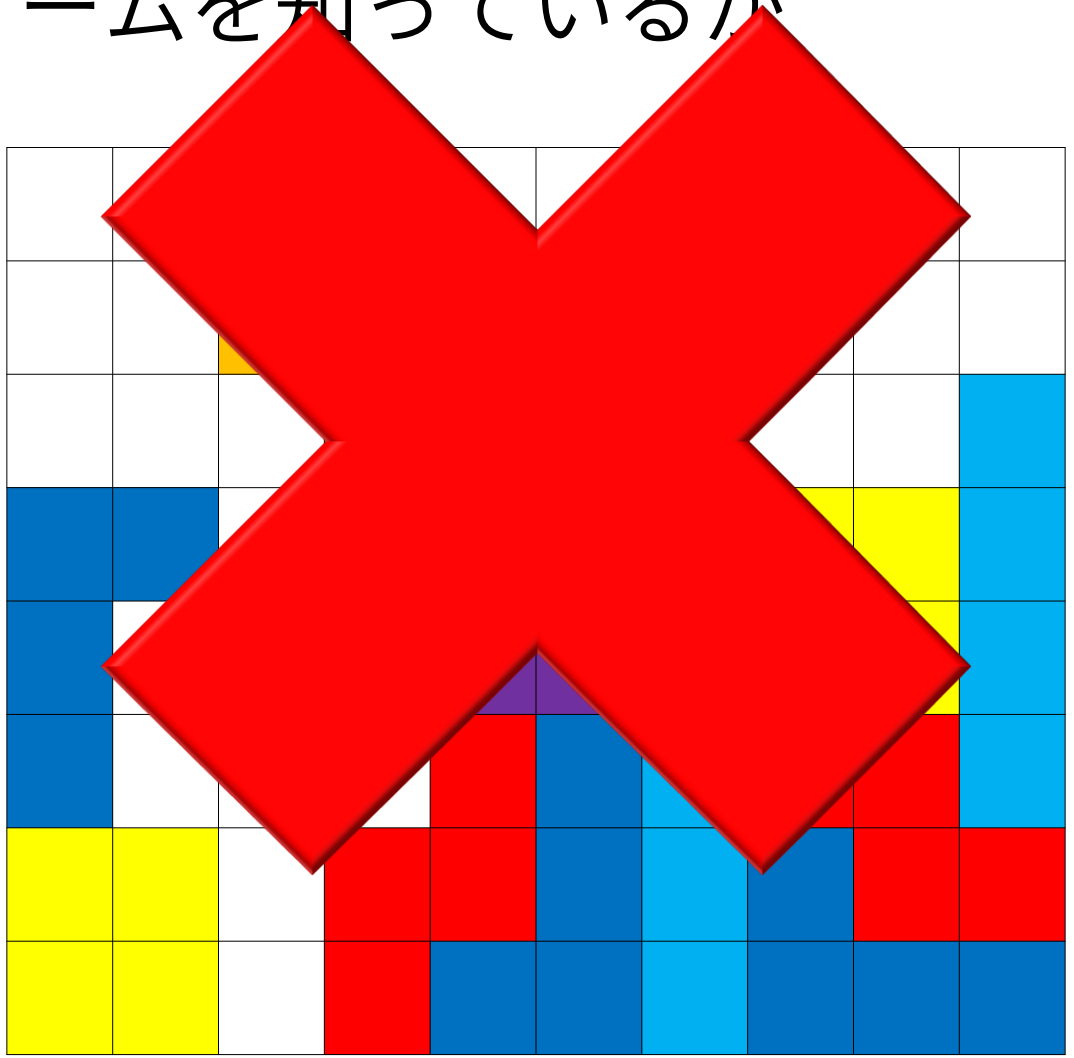
この問題の狙い

- テト〇スというゲームを知っているか



この問題の狙い

- テト〇スというゲームを知っているか



この問題の狙い

- Tricky T〇wersというゲームを知っているか

https://store.steampowered.com/app/437920/Tricky_Towers/?l=japanese

この問題の狙い

- Tricky T〇wersというゲームを知っているか

https://store.steampowe[illegible]om/app/43792[illegible]cky_Towers/?l=japanese

この問題の狙い

- DPを見抜けるか
  JOIをやや意識し、保持するデータ数の多い(DP配列の次元が大きい)DPを用意した

問題概要

- ゲームの得点が入ったログのデータが与えられる
- ログを満たす各ラウンドの対戦結果が有り得るか判定し、有り得るならばそれを出力

- プレイヤー数 : 4人
- ラウンド数N ≦ 100回

時間計算量からの推測

- プレイヤー数 : 4人
- ラウンド数N ≦ 100回

- O(N^4)なら間に合いそう
- 各プレイヤーについて何らかの情報を持たせてDPをすると良さそう

どんな情報を持つか

- 思いつくのは、
  - 今何ラウンド目を考えているか
  - プレイヤーAは何回得点を得ているか
  - プレイヤーBは何回得点を得ているか
  - プレイヤーCは何回得点を得ているか
  - プレイヤーDは何回得点を得ているか
- このままではN^{5}のデータが必要になる

どんな情報を持つか

- 思いつくのは、
  - 今何ラウンド目を考えているか
  - プレイヤーAは何回得点を得ているか
  - プレイヤーBは何回得点を得ているか
  - プレイヤーCは何回得点を得ているか
  - プレイヤーDは何回得点を得ているか
- このままではN^{5}のデータが必要になる？

どんな情報を持つか

- 思いつくのは、
  - 今何ラウンド目を考えているか
  - プレイヤーAは何回得点を得ているか・・・a
  - プレイヤーBは何回得点を得ているか・・・b
  - プレイヤーCは何回得点を得ているか・・・c
  - プレイヤーDは何回得点を得ているか・・・d
- 今何ラウンド目かは、(a+b+c+d)/3すれば求められる

- 他の要素から復元できる値がないか確認するのはDPにおいて非常に重要

実装面の注意

- $N^4=10^8$で、特にスクリプト型言語ではACするか微妙
- ループを回すDPよりも、再帰メモ化+枝切りの方が高速化が可能な場合が多い

- 例えば、a+b+c+dが3の倍数でない状況を満たす対戦結果は絶対に存在しない→処理が3分の1になる
- 特定のプレイヤーにのみ勝利/敗北が偏っている→後々厳しくなるのが目に見えている
- こういった工夫によって定数倍高速化が期待できる

実装面の注意

- $N^4=10^8$で、int型などでDPテーブルを持つとMLE
- bool型を使用する
- 入力を満たす対戦結果を出力するには、経路復元をする
- 経路復元はDP関数とほぼ同じように再帰で実装することができる

実装例

```
#include"bits/stdc++.h"
using namespace std;
int N, A, B, C, D, a[101]={}, b[101]={}, c[101]={}, d[101]={};
int main(){
		scanf("%d%d%d%d%d", &N, &A, &B, &C, &D);
		for(int i=0; i<A; i++) scanf("%d", a+i);
		for(int i=0; i<B; i++) scanf("%d", b+i);
		for(int i=0; i<C; i++) scanf("%d", c+i);
		for(int i=0; i<D; i++) scanf("%d", d+i);

		if((A+B+C+D)%3 == 0 && (A+B+C+D)/3 == N && dp(0, 0, 0, 0)){
				printf("%s¥n", "Yes");
				restore(0, 0, 0, 0);
		}
		else printf("%s¥n", "No");
}
```

- 関数「dp」「restore」は次のスライド

実装例

```
bool visited[101][101][101][101]={}, memo[101][101][101][101]={};
vector<int> v ={ 1, 2, 3 }, tmp;
bool dp(int i, int j, int k, int l){
        if(i==A && j==B && k==C && l==D) return true;
        if(visited[i][j][k][l]) return memo[i][j][k][l];
        tmp.clear();
        tmp ={ a[i], b[j], c[k] }; sort(tmp.begin(), tmp.end());
        if(v == tmp) memo[i][j][k][l] |= dp(i+1, j+1, k+1, l);
        tmp.clear();
        tmp ={ a[i], b[j], d[l] }; sort(tmp.begin(), tmp.end());
        if(v == tmp) memo[i][j][k][l] |= dp(i+1, j+1, k, l+1);
        tmp.clear();
        tmp ={ a[i], d[l], c[k] }; sort(tmp.begin(), tmp.end());
        if(v == tmp) memo[i][j][k][l] |= dp(i+1, j, k+1, l+1);
        tmp.clear();
        tmp ={ d[l], b[j], c[k] }; sort(tmp.begin(), tmp.end());
        if(v == tmp) memo[i][j][k][l] |= dp(i, j+1, k+1, l+1);
        return memo[i][j][k][l];
}
```

```
void restore(int i, int j, int k, int l){
        if(i==A && j==B && k==C && l==D) return;
        tmp.clear(); tmp ={ a[i], b[j], c[k] }; sort(tmp.begin(), tmp.end());
        if(v == tmp && dp(i+1, j+1, k+1, l)){
                printf("4¥n"); restore(i+1, j+1, k+1, l); return;
        }
        tmp.clear(); tmp ={ a[i], b[j], d[l] }; sort(tmp.begin(), tmp.end());
        if(v == tmp && dp(i+1, j+1, k, l+1)){
                printf("3¥n"); restore(i+1, j+1, k, l+1); return;
        }
        tmp.clear(); tmp ={ a[i], d[l], c[k] }; sort(tmp.begin(), tmp.end());
        if(v == tmp && dp(i+1, j, k+1, l+1)){
                printf("2¥n"); restore(i+1, j, k+1, l+1); return;
        }
        tmp.clear(); tmp ={ d[l], b[j], c[k] }; sort(tmp.begin(), tmp.end());
        if(v == tmp && dp(i, j+1, k+1, l+1)){
                printf("1¥n"); restore(i, j+1, k+1, l+1); return;
        }
}
```

Thank you for reading!