2022 校內賽初賽

題解

pA. 封弊者

封弊者: 高嘉泓

AC: 23人

出題者:
rurutoria7

題目概述

相信不會星爆的同學，是不會出現在這裡的吧？

pB.

FRANXX 配對

First AC: 張承漢

AC: 23 人

出題者:
chyen

題目概述

給定 2n 個數字,
問是否可以把他們兩兩分成一組使得同組數字的和為偶數

作答概述

- 檢查偶數數量
- 注意大小寫

子題

- 子題 2: 所有數不是全為奇數就是全為偶數
  很顯然答案永遠會是 Yes

- 子題 3 :
  紀錄偶數的數量, 若數量為偶數, 代表奇數的數量亦為偶數, 這時候答案為 Yes, 反之則為 No。

pC.

討伐天蠍座

First AC: 高嘉泓

AC: 6 人

出題者:
chyen

題目概述

- 把 n 個人分成 k 隊, 使每隊排名全距的最大值最小

作答概述

- sort + 二分搜

子題 2, 3

- 子題 2: $1 \leq n \leq 7$
  暴力枚舉每組人要被分到哪隊, 複雜度 O(n*k^n)

- 子題 3: $a_i = i$
  答案即為 ceil(n/k) -1

子題 4, 5

- 子題 4: $1 \leq n \leq 1000$
  將每個人的排名sort後排成一列後, 一段一段組成一隊會是最好的, 所以就sort完再去枚舉最大全距

- 子題 5:
  把枚舉最大全距的部分改成二分搜即可

參考解答　二分搜 check 的程式碼

```
bool check(int lim){
        int res = 0;
        for(int i=0;i<n;i*=1){
                int r = i;
                while(r<n && a[r]-a[i]<=lim)
                        r++;
                res++;
                i = r;
        }
        return res<=k;
}
```

AC code

完整連結

pD.

田野調查

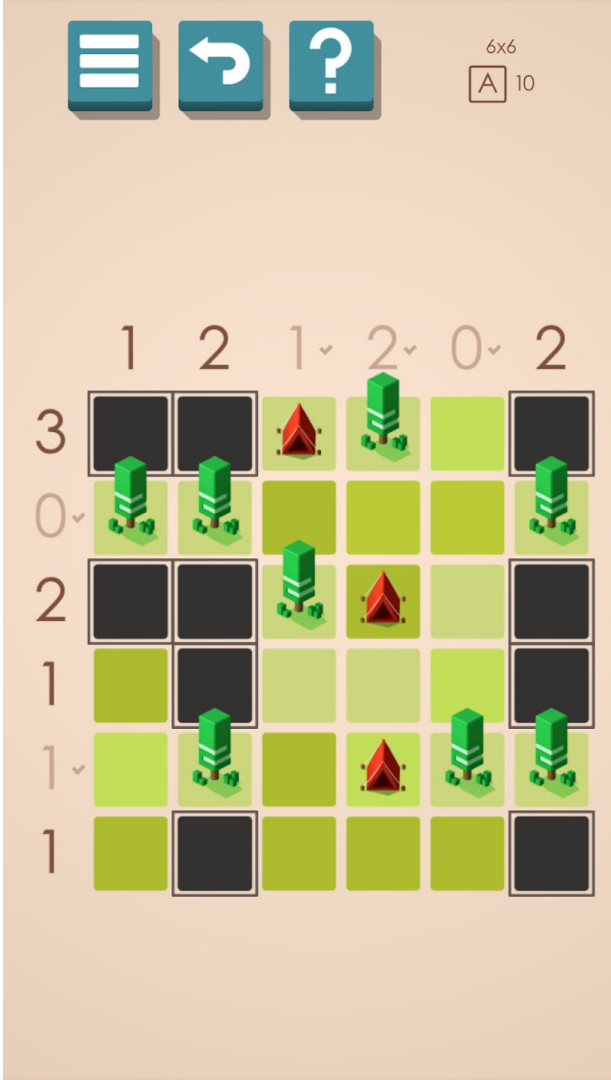
First AC: 高嘉泓

AC: 5 人

出題者:
xiplus

題目概述

- 就 Tents and Trees Puzzles 這個遊戲
- 沒有業配



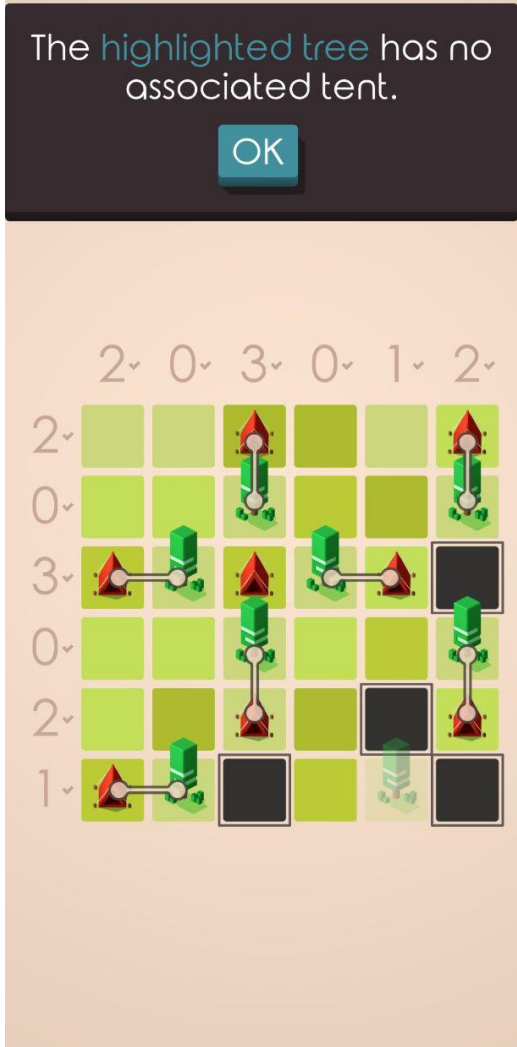
截圖自 Tents and Trees Puzzles

作答概述

- 實作題, DFS暴搜
- DFS by?
  - 每個空地要不要放帳篷? 複雜度 2^空地數
  - 每棵樹要綁哪個方向的帳篷? 複雜度 4^帳篷數
  - 範圍很小, 所以都可以, 但是...

作答概述

- 如果 DFS 空地，卻忘記判斷每棵樹是否「恰好」只綁到一座帳篷，就會出現右圖狀況
  - 如果你寫得出來，也可以
- DFS 樹，就不需要檢查這個，方便



截圖自 Tents and Trees Puzzles

子題

- 子題 2: 大小 2*2, 用 if 判一判
- 子題 3: 大小 1*m, 直接靠直行知道答案
  - 單純送分

參考解答

DFS 每棵樹要放哪個方向，並判斷其他條件有沒有衝突即可。

AC code

完整連結

```
for (int d1 = 0; d1 < 4; d1++) {
    int tr = trees[treei].first + dir4[d1].first;
    int tc = trees[treei].second + dir4[d1].second;
    if (!v[tr][tc] && cur_r[tr] < rcnt[tr] && cur_c[tc] < ccnt[tc]) {
        conflict = false;
        for (int d2 = 0; d2 < 8; d2++) {
            if (v[tr + dir8[d2].first][tc + dir8[d2].second] & TENT) {
                conflict = true;
                break;
            }
        }
        if (!conflict) {
            v[tr][tc] |= TENT;
            cur_r[tr]++;
            cur_c[tc]++;

            dfs(treei + 1);

            v[tr][tc] &= ~TENT;
            cur_r[tr]--;
            cur_c[tc]--;
        }
    }
}
```

pE.

超新手取向迷宮

First AC: 王淇

AC: 1 人

出題者:
ub33

題目概述

給一個N*M的迷宮，Q次詢問起點能不能走到終點。

作答概述

- dfs or dsu?

子題 2, 3

- 子題 2: N,M,Q<=100
  每次都從起點dfs, O(NMQ)

- 子題 3: N,M<=1000
  把每個連通塊著色, O(NM)

子題 4 詢問在左上和右下

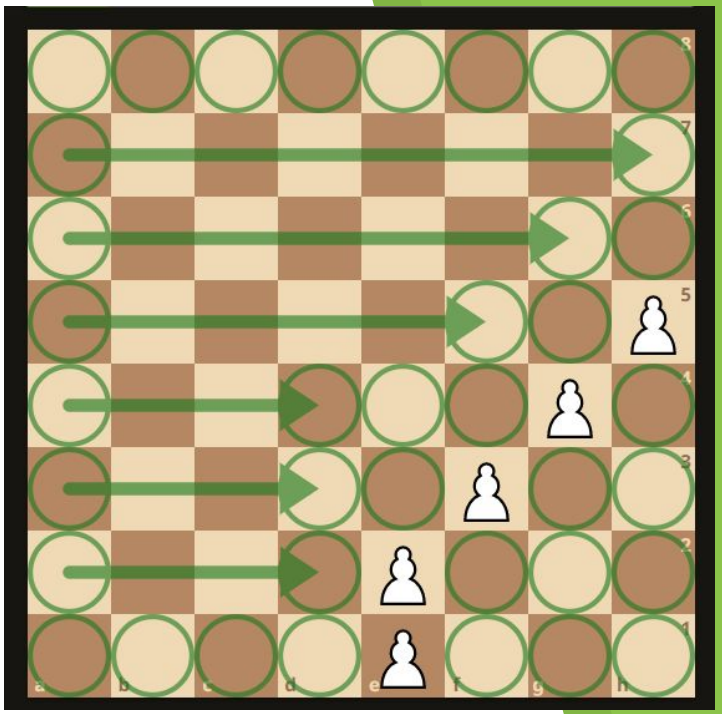
在牆壁上八方位走路, 如果牆壁能把起點和終點切開就是no

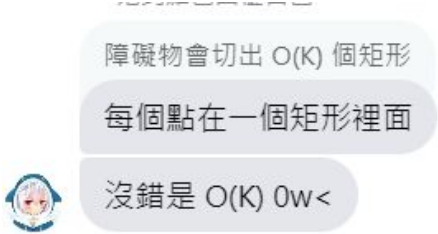
可以用map存圖

O(KlogK)

子題 5

原本預期解是常數超大的O(KlogK)
只建和牆壁八方位相鄰的點
dfs的時候另外找左右第一個碰的到的點
結果大家的解都比我還快, 很難過欸





AC code

完整連結

pF.

百鬼今天想要開台

First AC: 高嘉泓

AC: 2 人

出題者:
ub33

題目概述

每天可以選擇讓x=0或x=2x+a_i

求K天內不超過M有幾種方法

作答概述

- 矩陣快速冪

子題 1,2

- 子題 1
  暴搜

- 子題 2
  dp[前i天][現在的疲勞度]
  dp[i][x]->dp[i+1][2x+a_i]
  dp[i][x]->dp[i+1][0]
  O(nmk)

子題 3 n,m=1

把dp的狀態改成dp[i], 代表前i天且第i天休息有幾種方法

dp[i]=dp[i-1]+dp[i-2]

剛好費氏數列第k+2項

所以可以矩陣快速冪

$$\begin{bmatrix} f_{n+2} \\ f_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} f_{n+1} \\ f_n \end{bmatrix}$$

$$\begin{bmatrix} f_{n+1} \\ f_{n+2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n \begin{bmatrix} f_1 \\ f_0 \end{bmatrix}$$

子題 4 m<=100

把子題2的轉移式塞進矩陣即可

O(m^3 logk)

子題 5

把dp的狀態改成dp[i], 代表前i天且第i天休息有幾種方法
就變成dp[i]=Σdp[i-j]*(i到i-j這兩個休息之間可以塞幾種)
因為每次x都會變成兩倍以上, j最多大概logm
那坨東西可以O(nm logm)預處理
矩陣的大小也會變成logm
O(nmlogm + (logm)^3 logk)

AC code

完整連結

pG.

論拖延

First AC: 高嘉泓

AC: 2 人

出題者:
rurutoria7

題目概述

給你一張 n 點 m 邊的帶權無向圖,
以及起點 s、終點 t,
q 次詢問拔掉某條邊之後 s–t 最短路是否會變長。

作答概述

- 蓋出單源最短路DAG
- 求出 DAG 必經邊

子題

- 子題 2: 保證是一條鏈
- s 走到 t 的最短路徑, 只有一種方法。
- 每次詢問判斷該邊是否位於 s、t 之間即可

子題

- 子題 3: m=n-1
- 因為保證是一棵樹, 因此 s-t 最短路必定只有一條。
- 因此可以先從 s DFS,
  求出 st 路徑上有哪些邊, 預先存起來。
- 每次查詢檢查有沒有在其中即可

子題

- 子題 4: w=1 且 n,m <= 500
- 因為邊權為 1, 因此可以用 BFS 求出單源最短路。
- 先對原圖 BFS 一次, 求出 st 的最短路徑是多少。
- 對於每次的詢問, 都直接把邊拔掉, 重新 BFS 後比較原圖上的答案。
- 複雜度 $O(q(n+m))$

子題

- 子題 5: n,m <= 500
- 相較於上個子題, 拔掉了邊權為 1 的限制。
  只要將 BFS 改成 Dijkstra 即可。
- 複雜度 $O(q(n+m\log n))$

子題

- 子題 6: 無特殊限制

- 我們可以蓋出一張有向圖,
  包含所有從 s 到其他點的最短路徑。

- 由於 w>0, 因此這張圖是 DAG。
  這張 DAG 可以在 dijkstra 的時候蓋。

- 現在題目變成:回答 DAG 上的一條邊, 是否是 s 走到 t 的必經邊。

子題

- f(i,j) 是從 i 走到 j 的方案數。
  邊 (u,v) 是必經邊，若且唯若 f(s,u) * f(v,t) = f(s,t)。

- f(s,i) 在做 dijkstra 時可以順便蓋。f(j,t) 在蓋完 DAG 後，dp 算。

- 由於方案數很大&避免碰撞，多取幾個 mod。

AC code

完整連結

```
const int N = 2e5+10, MOD1 = 1e9+7, MOD2 = 1e8+7;

int n, m, s, t, q;

vector<int> G[N]; // {eid}
vector<pii> edg; // weight, to

typedef pii posb;
posb operator + (posb i, posb j)
{
    return { u1: (i.ff+j.ff)%MOD1,  u2: (i.ss+j.ss)%MOD2};
}
posb operator * (posb i, posb j)
{
    return { u1: (i.ff*j.ff)%MOD1,  u2: (i.ss*j.ss)%MOD2};
}
```

AC code

```
int dis[N];
posb sto[N];
void dijks() // s to t
{
    memset( b: dis,  c: 0x3f,  len: sizeof(dis));
    priority_queue<pii, vector<pii>, greater<pii>> pq;
    dis[s] = 0;
    sto[s] = { u1: 1, u2: 1};
    pq.push( v: { u1: 0,  &: s});

    while (!pq.empty())
    {
        int d = pq.top().ff;
        int u = pq.top().ss;
        pq.pop();

        if (dis[u] < d) continue;

        for (auto e : long long : G[u])
        {
            int w = edg[e].ff;
            int v = edg[e].ss;
            if (dis[v]>d+w)
            {
                sto[v] = sto[u];
                dis[v] = d+w;
                pq.push( v: { &: dis[v],  &: v});
            }
            else if (dis[v] == d+w)
            {
                sto[v] = sto[v] + sto[u];
            }
        }
    }
}
```

AC code

```
posb tot[N];
posb f(int u)
{
    if (tot[u].ff < 0)
    {
        if (u == t) tot[u] = { u1: 1, u2: 1};
        else
        {
            tot[u] = { u1: 0, u2: 0};
            for (auto e : long long : G[u])
            {
                int w = edg[e].ff;
                int v = edg[e].ss;
                if (dis[v] != dis[u]+w) continue;
                tot[u] = (tot[u] + f( u: v));
            }
        }
    }
    return tot[u];
}
```

AC code

```
signed main()
{
    lyx
    cin >> n >> m >> s >> t;
    for (int i=1; i<=m; i++)
    {
        int u, v, w;
        cin >> u >> v >> w;
        G[u].pb(edg.size());
        edg.pb({ &: w,  &: v});
        G[v].pb(edg.size());
        edg.pb({ &: w,  &: u});
    }

    dijks(); // build s-t dag and sto
    rep(i,1,n) tot[i] = { u1: -1, u2: -1};

    cin >> q;
    while (q--)
    {
        int x, u, v, w;
        cin >> x;
        u = edg[(x-1)*2].ss;
        v = edg[(x-1)*2+1].ss;
        w = edg[(x-1)*2].ff;
        if (dis[u] > dis[v])
            swap( &: u,  &: v);

        if (dis[u] + w != dis[v])
            cout << "no\n";
        else if (sto[u]*f( u: v) == sto[t])
            cout << "yes\n";
        else
            cout << "no\n";
    }
}
```