A - Multiple Subject Lessons

DP+组合数学 or 搜索

设为用了k种颜色，前j个数，和为i的方案数。

对于，因为大于的数不能选了，有

对于的情况，遍历当前的个数那么已知方案为，再对个进行染色，得到的方案数为，对当前的贡献为，于是得到状态转移方程

搜索也可以过

#include <cstdio>

#include <algorithm>

using namespace std;

int n, k1;

long long DP[20][20], c[35][35];

int main(){

for(int i = 0; i <= 30; ++i){

for(int j = 0; j <= i; ++j){

if(!j||!i) c[i][j] = 1;

else c[i][j] = c[i-1][j] + c[i-1][j-1];

}

}

scanf("%d %d",&n,&k1);

DP[0][0] = 1;

for(int i = 0 ; i <= n ; ++i){

for(int j = 1 ; j <= n ; ++j){

if(j>i){

DP[i][j] = DP[i][i];

continue;

}

for(int k = 0 ; j\*k <= i ; ++k){

DP[i][j] += DP[i-j\*k][j-1] \* c[k1+k-1][k];

}

}

}

printf("%lld\n",DP[n][n]);

return 0;

}

B - Chrismas Game

博弈

***Fusion principle***

对无向图做如下改动：将图中任意一个偶环缩成一个新点，任意一个奇环缩成一个新点加一个新边；所有连到原先环上的边全部与新点相连，这样的改动不会影响图的sg值。

由于图中的环最多与树有一个点连接，所以我们用Tarjan找出环直接删点即可。

最后整个游戏的sg值当做多个图上删边的子游戏的sg值的异或，当且仅当不等于0时为n-positon。

#include <cstdio>

#include <algorithm>

#include <stack>

using namespace std;

const int MAXN = 1e4 + 5;

int head[MAXN], tot;

struct Edge{

int to, next;

}G[MAXN];

inline void addEdge(int u, int v){

G[++tot].to = v;

G[tot].next = head[u];

head[u] = tot;

}

int n, m, k, sg[MAXN], dfn[MAXN], low[MAXN], id[MAXN], time, sum[MAXN], cnt, ans;

bool inStack[MAXN], del[MAXN];

stack<int> stk;

void init(){

for(int i = 1 ; i <= m ; ++i) head[i] = dfn[i] = low[i] = id[i] = sum[i] = inStack[i] = del[i] = 0;

tot = cnt = time = 0;

while(!stk.empty()) stk.pop();

}

inline void tarjan(int np, int fat){

stk.push(np);

inStack[np] = true;

dfn[np] = low[np] = ++time;

for(int i = head[np], to ; i ; i = G[i].next){

to = G[i].to;

if(to==fat) continue;

if(!dfn[to]){

tarjan(to,np);

low[np] = min(low[np],low[to]);

}else if(inStack[to]){

low[np] = min(low[np],low[to]);

}

}

if(low[np]==dfn[np]){

int k, last;

++cnt;

do{

last = k;

k = stk.top(); stk.pop();

inStack[k] = false;

++sum[id[k]=cnt];

if(np^k) del[k] = true;

if(np==k&&sum[cnt]&1&&sum[cnt]>1) del[last] = false; // 奇环保留一个点

}while(np^k);

}

}

int DFS(int np, int fat){

int res = 0;

for(int i = head[np], to ; i ; i = G[i].next){

to = G[i].to;

if(to==fat) continue;

if(!del[to]) res ^= DFS(to,np) + 1;

}

return res;

}

int main(){

while(scanf("%d",&n)!=EOF){

ans = 0;

for(int i = 1 ; i <= n ; ++i){

scanf("%d %d",&m,&k);

init();

for(int j = 1, u, v ; j <= k ; ++j){

scanf("%d %d",&u,&v);

addEdge(u,v);

addEdge(v,u);

}

tarjan(1,0);

ans ^= DFS(1,0);

}

puts(ans?"Sally":"Harry");

}

return 0;

}

C - Klee in Solitary Confinement

思维+模拟

注意到a的范围只有2e6，我们枚举操作对象数x，就可以得到x+k的最大出现次数，此时只需要考虑x与x+k组成的新数列，可由双指针得到。

令为在区间内x的出现次数，为在区间内x+k的出现次数，为新数列长度，假设当前修改区间为[l,r]则我们需要最大化

因

整理得

即对于每一个，最大化，记录前缀最大值转移即可。

#include <cstdio>

#include <vector>

#include <unordered\_set>

#include <algorithm>

using namespace std;

const int OFFSET = 1e6;

inline int read(){

int x = 0, f = 1;

char ch = getchar();

while(ch<'0'||ch>'9'){

if(ch=='-')

f = -1;

ch = getchar();

}

while(ch>='0'&&ch<='9'){

x=(x<<1)+(x<<3)+(ch^48);

ch = getchar();

}

return x\*f;

}

int n, k, ans, a[1000005], xy[1000005], prefix[1000005];

unordered\_set<int> st;

vector<int> idx[2000005];

int main(){

scanf("%d %d",&n,&k);

for(int i = 1 ; i <= n ; ++i){

a[i] = read();

idx[a[i]+OFFSET].emplace\_back(i);

st.insert(a[i]);

ans = max((int)idx[a[i]+OFFSET].size(),ans);

}

if(!k){

printf("%d\n",ans);

return 0;

}

for(int it : st){

int x = it, y = it + k, l1, l2, l, p1, p2;

if(y+OFFSET>2e6||y+OFFSET<0) l2 = 0;

else l2 = idx[y+OFFSET].size();

l1 = idx[x+OFFSET].size();

l = l1+l2;

prefix[0] = p1 = p2 = 0;

while(p1<l1||p2<l2){

if(p2>=l2&&p1<l1) xy[p1+p2+1]=x, ++p1;

else if(p1>=l1&&p2<l2) xy[p1+p2+1]=y, ++p2;

else if(idx[x+OFFSET][p1]<idx[y+OFFSET][p2]) xy[p1+p2+1]=x, ++p1;

else xy[p1+p2+1]=y, ++p2;

prefix[p1+p2] = prefix[p1+p2-1] + (xy[p1+p2]==y);

}

ans = max(ans,prefix[l]-prefix[1]+(xy[1]==x));

int mx = max(2\*prefix[1]-2,-1);

for(int r = 2 ; r <= l ; ++r){

ans = max(ans,mx+r+1-prefix[r]+prefix[l]-prefix[r]);

mx = max(mx,2\*prefix[r]-(r+1));

}

}

printf("%d\n",ans);

return 0;

}

D - Bacteria

构造

对于在第阶段

每个细菌的长度为，故有种可能出现的序列

一共有条序列。

综上第i阶段细菌的种类不大于和，前者递减，后者递增

当时这个部分最多有种。

若能使时这个部分有种，则不同的细菌数量达到上限。

找到最大的k且，构造序列保证第层满足一下两个条件

1. *条不同且长度为的序列*
2. 形成第层的细胞序列是各不相同的

条件①保证层后种类达到上限因为第层已经出现所有可能出现的序列

条件②保证层前种类达到上限

一种构造方法是在第k层填满0,1,2,…的长度为的二进制序列

#include <cstdio>

using namespace std;

int n, k;

bool ans[1<<21];

int main(){

scanf("%d",&n);

for(int i = 1 ; i <= n + 1; ++i)

if((1<<(n-i+1))>=i-1) k = i;

for(int i = 0 ; i < 1<<(k-1) ; ++i){

for(int j = 0 ; j < 1<<(n-k+1) ; ++j){

ans[(1<<(n-k+1))\*i+j+1] = i&(1<<j);

}

}

for(int i = 1 ; i <= 1<<n ; ++i)

printf("%d",ans[i]);

return 0;

}

E - SMS from MCHS

签到

#include <cstdio>

#include <iostream>

using namespace std;

int t1, v1, t2, v2;

int main(){

cin>>t1>>v1>>t2>>v2;

if(t2<0&&v2>=10)

puts("A storm warning for tomorrow! Be careful and stay home if possible!");

else if(t2<t1)

puts("MCHS warns! Low temperature is expected tomorrow.");

else if(v2>v1)

puts("MCHS warns! Strong wind is expected tomorrow.");

else

puts("No message");

return 0;

}

F – Garland

树形dp

题意：给一颗树，问能否把树分成和相等的三部分

首先判断全部和是否为3的倍数，不是的话无解

然后直接DFS找和为的子树删掉即可

#include <cstdio>

#include <vector>

using namespace std;

const int MAXN = 1e6 + 5;

int head[MAXN], tot;

struct Edge{

int to, next;

}G[MAXN<<1];

inline void addEdge(int u, int v){

G[++tot].to = v;

G[tot].next = head[u];

head[u] = tot;

}

int n, a[MAXN], sum, dp[MAXN];

vector<int> ans;

void DFS(int np, int fat = 0){

dp[np] = a[np];

int rec;

for(int i = head[np], to ; i ; i = G[i].next){

to = G[i].to;

if(to==fat) rec = (i+1)>>1;

else{

DFS(to,np);

dp[np] += dp[to];

}

}

if(dp[np]==sum/3)

ans.push\_back(rec), dp[np] = 0;

}

int main(){

scanf("%d",&n);

for(int i = 1, to ; i <= n ; ++i){

scanf("%d %d",&to,&a[i]);

if(to){

addEdge(to,i);

addEdge(i,to);

}else tot += 2;

sum += a[i];

}

if(sum%3){

puts("-1");

return 0;

}

DFS(1);

if(ans.size()>=3){

printf("%d %d\n",ans[0],ans[1]);

}else puts("-1");

return 0;

}

G – Hotel

线段树

线段树维护区间最大0长度，从左端开始最大0长度，从右端开始最大0长度。

#include <cstdio>

#include <algorithm>

using namespace std;

const int MAXN = 5e4 + 5;

struct node{

int lseg, seg, rseg, lazytag;

}tree[MAXN<<2];

int n, q;

void pushdown(int node, int l, int r){

if(tree[node].lazytag==-1) return ;

int left\_node = node<<1, right\_node = node<<1|1, mid = (l+r)>>1;

tree[left\_node].lazytag = tree[right\_node].lazytag = tree[node].lazytag;

tree[left\_node].lseg = tree[left\_node].seg = tree[left\_node].rseg = (mid-l+1)\*!tree[node].lazytag;

tree[right\_node].lseg = tree[right\_node].seg = tree[right\_node].rseg = (r-mid)\*!tree[node].lazytag;

tree[node].lazytag = -1;

}

void build(int node, int l, int r){

tree[node].lseg = tree[node].seg = tree[node].rseg = r - l + 1;

tree[node].lazytag = -1;

if(l==r) return ;

int left\_node = node<<1, right\_node = node<<1|1, mid = (l+r)>>1;

build(left\_node,l,mid);

build(right\_node,mid+1,r);

}

int find(int node, int l, int r, int val){

if(l==r) return l;

int left\_node = node<<1, right\_node = node<<1|1, mid = (l+r)>>1;

pushdown(node,l,r);

if(tree[left\_node].seg>=val) return find(left\_node,l,mid,val);

else if(tree[left\_node].rseg+tree[right\_node].lseg>=val) return mid - tree[left\_node].rseg + 1;

else if(tree[right\_node].seg>=val) return find(right\_node,mid+1,r,val);

return 0;

}

int cnt;

void modify(int node, int l, int r, int s, int e, int val){

if(r<s||l>e) return ;

if(l>=s&&r<=e){

tree[node].lazytag = val;

tree[node].lseg = tree[node].seg = tree[node].rseg = (r-l+1)\*!val;

return ;

}

if(r-l) pushdown(node,l,r);

int left\_node = node<<1, right\_node = node<<1|1, mid = (l+r)>>1;

modify(left\_node,l,mid,s,e,val);

modify(right\_node,mid+1,r,s,e,val);

tree[node].seg = max(max(tree[right\_node].seg,tree[left\_node].seg),tree[left\_node].rseg+tree[right\_node].lseg);

if(tree[left\_node].seg==mid-l+1) tree[node].lseg = mid-l+1+tree[right\_node].lseg;

else tree[node].lseg = tree[left\_node].lseg;

if(tree[right\_node].seg==r-mid) tree[node].rseg = r-mid+tree[left\_node].rseg;

else tree[node].rseg = tree[right\_node].rseg;

}

int main(){

scanf("%d %d",&n,&q);

build(1,1,n);

for(int i = 1, opt, a, b ; i <= q ; ++i){

scanf("%d",&opt);

if(opt==1){

scanf("%d",&a);

printf("%d\n",b=find(1,1,n,a));

if(b) modify(1,1,n,b,b+a-1,1);

}else{

scanf("%d %d",&a,&b);

modify(1,1,n,a,min(a+b-1,n),0);

}

}

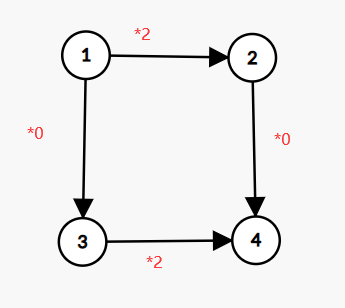
return 0;

}

H - Minimum path

分层图+最短路

先考虑将最短路上任意一条边的权重乘2和任意一条边权重归0



在此分层图上跑最短路

由于最短路的特性，答案就是图4的dis且乘2的必然是权重最小的边，乘0的必然是权重最大的边

#include <iostream>

#include <algorithm>

#include <queue>

#include <cstring>

using namespace std;

typedef long long ll;

typedef unsigned long long ull;

typedef pair<ll,int> node;

const int INF = 1e9;

const int MAXN = 200005;

int head[MAXN<<2], tot;

struct Edge{

int to, val, next;

}G[MAXN\*18];

inline void addEdge(int u, int v, int val){

G[++tot].to = v;

G[tot].val = val;

G[tot].next = head[u];

head[u] = tot;

}

int n, m;

ll dis[MAXN<<2];

bool vis[MAXN<<2];

inline void ADD(int s, int t, int v){

// 1

addEdge(s,t,v);

addEdge(s+n,t+n,v);

addEdge(s+(n<<1),t+(n<<1),v);

addEdge(s+n\*3,t+n\*3,v);

// addEdge(s,t+n\*3,v);

// 1 -max-> 2 -min-> 4

addEdge(s,t+n,v<<1);

addEdge(s+n,t+n\*3,0);

// 1 -min-> 3 -max-> 4

addEdge(s,t+(n<<1),0);

addEdge(s+(n<<1),t+n\*3,v<<1);

}

void dijkstra(int s){

memset(dis,0x3f,sizeof(dis));

priority\_queue<node,vector<node>,greater<node> > pq;

pq.push(node{dis[s]=0,s});

while(!pq.empty()){

int np = pq.top().second; pq.pop();

if(vis[np]) continue;

vis[np] = true;

for(int i = head[np], to ; i ; i = G[i].next){

to = G[i].to;

if(!vis[to]&&dis[to]>dis[np]+G[i].val){

dis[to]=dis[np]+G[i].val;

pq.push(node(dis[to],to));

}

}

}

}

int main(){

scanf("%d %d",&n,&m);

for(int i = 1, u, v, w ; i <= m ; ++i){

scanf("%d %d %d",&u,&v,&w);

ADD(u,v,w);

ADD(v,u,w);

}

dijkstra(1);

for(int i = 3 \* n + 2 ; i <= (n<<2) ; ++i){

printf("%lld%c",min(dis[i],dis[i-3\*n])," \n"[i==n<<2]);

}

return 0;

}