搜索

A*

k 短路

```
#include<bits/stdc++.h>
using namespace std;

#define x first
#define y second
typedef pair<int,int> PII;
typedef pair<int,pair<int,int>> PIII;

const int N=1005, M=2e5+5;
const int INF=0x3f3f3f3f;
int n,m,s,t,k;

int head[N],rhead[N],tot;
struct node{
    int to,next,w;
}e[M];

void add(int h[],int u,int v,int w)
{e[tot].to=v;e[tot].w=w;e[tot].next=h[u];h[u]=tot++;}

int d[N];
bool vis[N];

void dijk(){
    memset(d,0x3f,sizeof d);
    priority_queue<PII,vector<PII>,greater<PII>> q;

    q.push({0,t});
    d[t]=0;

    while(q.size()){
        auto hd=q.top(); q.pop();

        int ver=hd.y;
        if(vis[ver]) continue;
        vis[ver]=true;
```

```
        for(int i=rhead[ver];~i;i=e[i].next){
            int go=e[i].to;
            if(d[go]>d[ver]+e[i].w){
                d[go]=d[ver]+e[i].w;
                q.push({d[go],go});
            }
        }
    }
}

int astar(){
    priority_queue<PIII,vector<PIII>,greater<PIII>> q;
    int cnt=0;

    if(d[s]==INF) return -1;
    q.push({d[s],{0,s}});

    while(q.size()){
        auto hd=q.top(); q.pop();

        int ver=hd.y.y;
        int dis=hd.y.x;
        if(ver==t) cnt++;
        if(cnt==k) return dis;

         for(int i=head[ver];~i;i=e[i].next){
            int go=e[i].to;
            q.push({dis+e[i].w+d[go],{dis+e[i].w,go}});
         }
    }

    return -1;
}

int main(){
    memset(head,-1,sizeof head);
    memset(rhead,-1,sizeof rhead);

    cin>>n>>m;
    while(m--){
        int u, v, w; cin>>u>>v>>w;
        add(head,u,v,w); add(rhead,v,u,w);
    }
    cin>>s>>t>>k;
    if(s==t) k++;

    dijk(); // for the g

    cout<<astar()<<endl;

    return 0;
}
```

IDA*

```
// Problem: 排书
// Contest: AcWing
// URL: https://www.acwing.com/problem/content/182/
// Memory Limit: 64 MB
// Time Limit: 1000 ms
//
// Powered by CP Editor (https://cpeditor.org)

#include<bits/stdc++.h>
using namespace std;

#define debug(x) cerr << #x << ": " << (x) << endl
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define dwn(i,a,b) for(int i=(a);i>=(b);i--)

using pii = pair<int, int>;
using ll = long long;

inline void read(int &x){
    int s=0; x=1;
    char ch=getchar();
    while(ch<'0' || ch>'9') {if(ch=='-')x=-1;ch=getchar();}
    while(ch>='0' && ch<='9') s=(s<<3)+(s<<1)+ch-'0',ch=getchar();
    x*=s;
}

const int N=20;

int w[5][N], q[N];
int n;

bool ok(){
    rep(i,1,n) if(q[i]!=i) return false;
    return true;
}

int f(){
    int cnt=0;
    rep(i,1,n-1) if(q[i]!=q[i+1]-1) cnt++;
    return (cnt+2)/3;
}

bool dfs(int cur, int lim){
    if(cur+f()>lim) return false;
    if(ok()) return true;

    rep(l,1,n) rep(r,l,n){
        for(int k=1; k+r-l<=n; k++){
            memcpy(w[cur], q, sizeof q);

            rep(i,k,k+r-l) q[i]=w[cur][l+i-k];
            for(int i=1, j=1; i<=n; i++){
                while(j>=l && j<=r) j++;
                if(i>=k && i<=k+r-l) continue;
                q[i]=w[cur][j++];
```

```
            }

            if(dfs(cur+1, lim)) return true;
            memcpy(q, w[cur], sizeof w[cur]);
        }
    }
    return false;
}

int main(){
    int T; cin>>T;
    while(T--){
        cin>>n;
        rep(i,1,n) read(q[i]);

        int dep=0;
        while(dep<5 && !dfs(0, dep)) dep++;
        if(dep==5) puts("5 or more");
        else cout<<dep<<endl;
    }
    return 0;
}
```

DLX

精确覆盖问题

给定一个 $N \times M$ 的数字矩阵 A，矩阵中的元素 $Ai,j \in \{0,1\}$。

请问，你能否在矩阵中找到一个行的集合，使得这些行中，每一列都有且仅有一个数字 1。

```
const int M=505;
const int N=M*(M+1); // 1 的个数

int n, m;
int w[M][M];

struct DLX{
    int l[N], r[N], u[N], d[N]; // 四个指针
    int s[N]; // s[j] 表示第 j 列 1 的数量
    int row[N], col[N]; // 相应标号所在行、列
    int idx; // 点的标号

    int ans[N], top;

    void init(){ // 一开始有 m+1 个点
        rep(i,0,m) l[i]=i-1, r[i]=i+1, u[i]=d[i]=i;
        l[0]=m, r[m]=0;
        idx=m+1;
    }

    void add(int &hh, int &tt, int x, int y){
        row[idx]=x, col[idx]=y, s[y]++;
        u[idx]=y, d[idx]=d[y], u[d[y]]=idx, d[y]=idx;
```

```
        r[hh]=l[tt]=idx, r[idx]=tt, l[idx]=hh;
        tt=idx++;
    }

    void build(){
        rep(i,1,n){
            int hh=idx, tt=idx;
            rep(j,1,m){
                int x=w[i][j];
                if(x) add(hh, tt, i, j);
            }
        }
    }

    void remove(int p){
        r[l[p]]=r[p], l[r[p]]=l[p];
        for(int i=d[p]; i!=p; i=d[i]) for(int j=r[i]; j!=i; j=r[j])
            s[col[j]]--, u[d[j]]=u[j], d[u[j]]=d[j];
    }

    void resume(int p){
        for(int i=u[p]; i!=p; i=u[i]) for(int j=l[i]; j!=i; j=l[j])
            u[d[j]]=j, d[u[j]]=j, s[col[j]]++;
        r[l[p]]=p, l[r[p]]=p;
    }

    bool dfs(){
        if(!r[0]) return true;
        int p=r[0];
        for(int i=r[0]; i; i=r[i]) if(s[i]<s[p]) p=i; // 找到点最少的列
        remove(p);
        for(int i=d[p]; i!=p; i=d[i]){
            ans[++top]=row[i];
            for(int j=r[i]; j!=i; j=r[j]) remove(col[j]);
            if(dfs()) return true;
            for(int j=l[i]; j!=i; j=l[j]) resume(col[j]);
            top--;
        }
        resume(p);
        return false;
    }

    void solve(){
        if(dfs()){
            rep(i,1,top) cout<<ans[i]<<' ';
            puts("");
        }
        else puts("No Solution!");
    }
}dlx;

int main(){
    cin>>n>>m;
    rep(i,1,n) rep(j,1,m) read(w[i][j]);

    dlx.init();
    dlx.build();
    dlx.solve();
```

```
    return 0;
}
```

例题：数独

```
/*
 * DLX解经典数独问题
 * 考虑到每行都要有1~9
 * 用第1~81列表示第i行是否有数字j(第(i - 1)*9 + j列的状态）
 * 由于每列都要有1~9
 * 用第82~162列表示第i列是否有数字j (第(i - 1)*9 + j + 81列的状态)
 * 考虑到每个九宫格都要有1~9
 * 用第163~243列表示第i个九宫格是否有数字j (第(i - 1)*9 + j + 162列的状态)
 * 但是仅仅依靠这些是不够的
 * 还有一个限制条件每个格子只能有1个数字
 * 所以用第244~324列表示格子中是否填了数
 * 那么对于给定的数字，就在对应的列上写4个1
 * 如果是.那么就添加9行，分别对应哪个位子填1~9的情况
 * 那么最坏情况下有9*9*9行可选
 */

const int R=9*9*9+5, C=9*9*4+5;
const int N=4*R; // 1 的个数

int n, m;
int w[R][C];

struct DLX{
    int l[N], r[N], u[N], d[N]; // 四个指针
    int s[N]; // s[j] 表示第 j 列 1 的数量
    int row[N], col[N]; // 相应标号所在行、列
    int idx; // 点的标号

    int ans[N], top;
    int res[10][10];

    void init(){ // 一开始有 m+1 个点
        memset(row, 0, sizeof row);
        memset(col, 0, sizeof col);
        memset(s, 0, sizeof s);
        top=0;
        rep(i,0,m) l[i]=i-1, r[i]=i+1, u[i]=d[i]=i;
        l[0]=m, r[m]=0;
        idx=m+1;
    }

    void add(int &hh, int &tt, int x, int y){
        row[idx]=x, col[idx]=y, s[y]++;
        u[idx]=y, d[idx]=d[y], u[d[y]]=idx, d[y]=idx;
        r[hh]=l[tt]=idx, r[idx]=tt, l[idx]=hh;
        tt=idx++;
    }

    void build(){
        rep(i,1,n){
```

```
            int hh=idx, tt=idx;
            rep(j,1,m){
                int x=w[i][j];
                if(x) add(hh, tt, i, j);
            }
        }
    }

    void remove(int p){
        r[l[p]]=r[p], l[r[p]]=l[p];
        for(int i=d[p]; i!=p; i=d[i]) for(int j=r[i]; j!=i; j=r[j])
            s[col[j]]--, u[d[j]]=u[j], d[u[j]]=d[j];
    }

    void resume(int p){
        for(int i=u[p]; i!=p; i=u[i]) for(int j=l[i]; j!=i; j=l[j])
            u[d[j]]=j, d[u[j]]=j, s[col[j]]++;
        r[l[p]]=p, l[r[p]]=p;
    }

    bool dfs(){
        if(!r[0]) return true;
        int p=r[0];
        for(int i=r[0]; i; i=r[i]) if(s[i]<s[p]) p=i; // 找到点最少的列
        remove(p);
        for(int i=d[p]; i!=p; i=d[i]){
            ans[++top]=row[i];
            for(int j=r[i]; j!=i; j=r[j]) remove(col[j]);
            if(dfs()) return true;
            for(int j=l[i]; j!=i; j=l[j]) resume(col[j]);
            top--;
        }
        resume(p);
        return false;
    }

    void solve(){
        if(dfs()){
            rep(i,1,top){
                int t=ans[i];
                int r=(t-1)/9/9+1, c=(t-1)/9%9+1, v=(t-1)%9+1;
                res[r][c]=v;
            }

            rep(i,1,9) rep(j,1,9) cout<<res[i][j];
            cout<<endl;
        }
        // else puts("No Solution!");
    }
}dlx;

int get(int x, int y){
    int r=(x-1)/3+1, c=(y-1)/3+1;
    return (r-1)*3+c;
}

int main(){
    n=9*9*9, m=4*9*9;
```

```
    string s;
    while(cin>>s, s!="end"){
        int p=0;

        vector<pii> buf;

        rep(i,1,9) rep(j,1,9){
            char t=s[p++];
            int val=(t=='.'? 0: t-'0');
            rep(k,1,9){
                if(val && val!=k) continue;
                int r=(i-1)*9*9+(j-1)*9+k;
                int c1, c2, c3, c4;
                c1=(i-1)*9+j;
                c2=9*9+(i-1)*9+k;
                c3=2*9*9+(j-1)*9+k;
                c4=3*9*9+(get(i, j)-1)*9+k;

                w[r][c1]=w[r][c2]=w[r][c3]=w[r][c4]=1;
                buf.push_back({r, c1});
                buf.push_back({r, c2});
                buf.push_back({r, c3});
                buf.push_back({r, c4});
            }
        }

        dlx.init();
        dlx.build();
        dlx.solve();

        for(auto [x, y]: buf) w[x][y]=0;
    }
    return 0;
}
```

重复覆盖问题

给定一个 $N \times M$ 的数字矩阵 A，矩阵中的元素 $Ai,j \in \{0,1\}$。

请你在矩阵中找到一个行的集合，使得这些行中，每一列都包含数字 1，并且集合中包含的行数尽可能少。

```
const int N = 10010;

int n, m;
int l[N], r[N], u[N], d[N], col[N], row[N], s[N], idx;
int ans[N];
bool st[110];

void init()
{
    for (int i = 0; i <= m; i ++ )
```

```
    {
        l[i] = i - 1, r[i] = i + 1;
        col[i] = u[i] = d[i] = i;
        s[i] = 0;
    }
    l[0] = m, r[m] = 0;
    idx = m + 1;
}

void add(int& hh, int& tt, int x, int y)
{
    row[idx] = x, col[idx] = y, s[y] ++ ;
    u[idx] = y, d[idx] = d[y], u[d[y]] = idx, d[y] = idx;
    r[hh] = l[tt] = idx, r[idx] = tt, l[idx] = hh;
    tt = idx ++ ;
}

int h()
{
    int cnt = 0;
    memset(st, 0, sizeof st);
    for (int i = r[0]; i; i = r[i])
    {
        if (st[col[i]]) continue;
        cnt ++ ;
        st[col[i]] = true;
        for (int j = d[i]; j != i; j = d[j])
            for (int k = r[j]; k != j; k = r[k])
                st[col[k]] = true;
    }
    return cnt;
}

void remove(int p)
{
    for (int i = d[p]; i != p; i = d[i])
    {
        r[l[i]] = r[i];
        l[r[i]] = l[i];
    }
}

void resume(int p)
{
    for (int i = u[p]; i != p; i = u[i])
    {
        r[l[i]] = i;
        l[r[i]] = i;
    }
}

bool dfs(int k, int depth)
{
    if (k + h() > depth) return false;
    if (!r[0]) return true;
    int p = r[0];
    for (int i = r[0]; i; i = r[i])
        if (s[p] > s[i])
```

```
            p = i;

    for (int i = d[p]; i != p; i = d[i])
    {
        ans[k] = row[i];
        remove(i);
        for (int j = r[i]; j != i; j = r[j]) remove(j);
        if (dfs(k + 1, depth)) return true;
        for (int j = l[i]; j != i; j = l[j]) resume(j);
        resume(i);
    }
    return false;
}

int main()
{
    scanf("%d%d", &n, &m);
    init();
    for (int i = 1; i <= n; i ++ )
    {
        int hh = idx, tt = idx;
        for (int j = 1; j <= m; j ++ )
        {
            int x;
            scanf("%d", &x);
            if (x) add(hh, tt, i, j);
        }
    }

    int depth = 0;
    while (!dfs(0, depth)) depth ++ ;
    printf("%d\n", depth);
    for (int i = 0; i < depth; i ++ ) printf("%d ", ans[i]);
    return 0;
}
```

模拟退火

模拟退火求最小值

```
#include<bits/stdc++.h>
using namespace std;

#define x first
#define y second
typedef pair<double, double> PDD;

const int N=105;
PDD p[N];
int n;
double ans=1e9;

double rand(double l, double r){
    return (double)rand()/RAND_MAX*(r-l)+l;
```

```
}

double getd(PDD u, PDD v){
    double dx=u.x-v.x, dy=u.y-v.y;
    return sqrt(dx*dx+dy*dy);
}

double calc(PDD pt){
    double res=0;
    for(int i=1; i<=n; i++) res+=getd(pt, p[i]);
    ans=min(ans, res);
    return res;
}

void anneal(){
    PDD cur(rand(0, 1e4), rand(0, 1e4));
    for(double t=1e4; t>1e-4; t*=0.99){
        PDD np(rand(cur.x-t, cur.x+t), rand(cur.y-t, cur.y+t));
        double dt=calc(np)-calc(cur);
        if(exp(-dt/t)>rand(0, 1)) cur=np;
    }
}

int main(){
    cin>>n;
    for(int i=1; i<=n; i++){
        double x, y; cin>>x>>y;
        p[i]={x, y};
    }

    // while ((double)clock()/CLOCKS_PER_SEC<0.80) anneal();
    for(int i=0; i<100; i++) anneal();
    printf("%.0lf", ans);

    return 0;
}
```

模拟退火求最大值

```
#include<bits/stdc++.h>
using namespace std;

#define x first
#define y second
typedef pair<int, int> PII;

const int N=55;

int n, m;
PII p[N];

int ans;

int calc(){
    int res=0;
    for(int i=0; i<m; i++){
        res+=p[i].x+p[i].y;
```

```
        if(i==m-1) continue;

        if(p[i].x==10) res+=p[i+1].x+p[i+1].y;
        else if(p[i].x+p[i].y==10) res+=p[i+1].x;
    }
    ans=max(ans, res);
    return res;
}

void anneal(){
    bool ok=m-n; // 是否加了一场
    for(double t=1e4; t>1e-4; t*=0.99){
        int u=rand()%m, v=rand()%n;
        int cur=calc();
        swap(p[u], p[v]);

        if(p[n-1].x==10 && !ok || p[n-1].x!=10 && ok) swap(p[u], p[v]); // 不合法
情况
        else{
            int np=calc();
            int delta=np-cur;
            if(exp(delta/t)<(double)rand()/RAND_MAX) swap(p[u], p[v]);  // 跳不过
去
        }
    }
}

int main(){
    cin>>n;
    for(int i=0; i<n; i++) cin>>p[i].x>>p[i].y;
    if(p[n-1].x==10) m=n+1, cin>>p[n].x>>p[n].y;
    else m=n;

    for(int i=0; i<100; i++) anneal();
    cout<<ans<<endl;

    return 0;
}
```