2025-2026 秋《数据结构与算法》第四次实验题解

助教-陈孙一硕

Difficulties:

- Easy: A,B,C,D
- Medium-Easy: E
- Medium-Hard: F
- Hard: G

A 求无向图连通分量的数量

模板题。用邻接表来建图，DFS 搜索即可。

```
//#include<bits/stdc++.h>
#include<iostream>
#include<iomanip>
#include<vector>
#include<string>
#include<cmath>
#include<algorithm>
#include<set>
#include<map>
#include<unordered_map>
#include<unordered_set>
#include<queue>
#include<stack>
using namespace std;
#define int long long
#define endl '\n'
```

```
//const int mod=1e9+7;
const int mod=998244353;
const int N=2e6+101;
vector<int> g[N];
bool vis[N];
void dfs(int u) {
    vis[u] = true;
    for (auto v : g[u]) {
        if (!vis[v]) {
            dfs(v);
        }
    }
}
void solve() {
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; i++) {
        g[i].clear();
        vis[i] = false;
    }
    for (int i = 0; i < m; i++) {
        int u, v;
        cin >> u >> v;
        g[u].push_back(v);
        g[v].push_back(u);
    }
    int cnt = 0;
    for (int i = 1; i <= n; i++) {
        if (!vis[i]) {
            dfs(i);
            cnt++;
        }
    }
    cout << cnt << endl;
}
signed main() {
```

```
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int T = 1; // cin >> T;
    while (T--) solve();
    return 0;
}
```

B Captain-G's "Die Hard"

用 DFS 搜索固然可行。但这题有一个有意思的性质：对于某个值 w，只要满足 $w \leqslant X$，且 w 是 $\gcd(x, y)$ 的倍数，就一定是可行的，否则不行。发现本结论可快速通过本题。

下面证明这个结论：记 $d = \gcd(X, Y), \forall w = td, w \leqslant X$ 均有解。

必要性：首先证明，对于任意一个状态 (a, b)，均有 $d|a$ 且 $d|b$。

考虑数学归纳法：对于任意一个状态 (a, b) 满足 $d|a$ 且 $d|b$，有以下转移：

- 倒空第一个杯子，变为 $(0, b)$，显然成立
- 倒空第二个杯子，变为 $(a, 0)$，显然成立
- 倒满第一个杯子，变为 (X, b)，显然成立
- 倒满第二个杯子，变为 (a, Y)，显然成立
- 将第一个杯子倒入第二个杯子，此时讨论以下两种情况：
  - 第一个杯子中的水倒完了，变为 $(0, a + b)$，显然成立
  - 第一个杯子中的水有剩余，变为 $(a + b - Y, Y)$，显然成立
- 将第二个杯子倒入第一个杯子，由对称性同理可证明

于是得出，对于所有可达的 w，都一定是 d 的倍数。

下面证明所有小于等于 X 的 d 的倍数都是可达的。

充分性：

考虑 $X = dm, Y = dn, \gcd(m, n) = 1$。不妨设 $X \geqslant Y$。我们考虑以下构造：将 Y 装满水然后倒入 X，如果 X 满，则将 X 倒空之后再将剩余部分倒入。重复 t 次上述操作，显然最后 X 壶中剩余的水量为 $tY\%X = tdn\%dm = d(tn\%m)$。

那么我们只需要证明，在 t 充分大的时候，$tn\%m$ 能够取遍 $[0, m - 1]$ 上所有的值即可。

设 $r_t = tn\%m$，假设存在 $0 \leqslant t_1 < t_2 \leqslant m-1$ 使得 $t_1 n \equiv t_2 n \pmod m$，那么有 $(t_2 - t_1)n \equiv 0 \pmod m$。由于 $\gcd(m, n) = 1$，所以 $t_2 - t_1 \equiv 0 \pmod m$。然而 $t_2 - t_1 \leqslant m-1$，矛盾！因此在 $t = 0, 1, \ldots, m-1$ 时，$tn\%m$ 的值两两不同，即取遍了 $[0, m-1]$ 上的所有整数。

综上，证明了上述结论。

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int X, Y;
    cin >> X >> Y;
    int K;
    cin >> K;
    int g = gcd(X, Y);
    for (int i = 0; i < K; i++) {
        int w;
        cin >> w;
        if (w <= X && w % g == 0) cout << 1 << " ";
        else cout << 0 << " ";
    }
    return 0;
}
```

C 拆地毯

最小生成树模板题。这里用 Kruskal 算法实现。

```
//#include<bits/stdc++.h>
#include<iostream>
#include<iomanip>
#include<vector>
#include<string>
#include<cmath>
#include<algorithm>
#include<set>
#include<map>
```

```
#include<unordered_map>
#include<unordered_set>
#include<queue>
#include<stack>
using namespace std;
#define int long long
#define endl '\n'
//const int mod=1e9+7;
const int mod=998244353;
const int N=2e6+101;
struct Edge {
    int u, v, w;
    bool operator<(const Edge& other) const {
        return w > other.w;
    }
};
int parent[N];
int find(int x) {
    if(parent[x] != x) {
        parent[x] = find(parent[x]);
    }
    return parent[x];
}
bool unite(int x, int y) {
    x = find(x);
    y = find(y);
    if(x == y) return false;
    parent[y] = x;
    return true;
}
void solve() {
    int n, m, K;
    cin >> n >> m >> K;
    vector<Edge> edges(m);
    for(int i = 0; i < m; i++) {
        cin >> edges[i].u >> edges[i].v >> edges[i].w;
```

```
    }
    for(int i = 1; i <= n; i++) {
        parent[i] = i;
    }
    sort(edges.begin(), edges.end(), [](const Edge& a, const Edge& b) {
        return a.w > b.w;
    });
    int cnt = 0;
    int ans = 0;
    for(auto& e : edges) {
        if(cnt >= K) break;
        if(find(e.u) != find(e.v)) {
            unite(e.u, e.v);
            ans += e.w;
            cnt++;
        }
    }
    cout << ans << endl;
}
signed main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int T = 1; // cin >> T;
    while(T--) solve();
    return 0;
}
```

D 我爱首都

Floyd 算法模板题。

```
//#include<bits/stdc++.h>
#include<iostream>
#include<iomanip>
#include<vector>
```

```
#include<string>
#include<cmath>
#include<algorithm>
#include<set>
#include<map>
#include<unordered_map>
#include<unordered_set>
#include<queue>
#include<stack>
using namespace std;
#define int long long
#define endl '\n'
//const int mod=1e9+7;
const int mod=998244353;
const int N=2e6+101;
const int INF=1e18;
int dist[110][110];
void solve(){
    int n, m;
    cin >> n >> m;
    for(int i=1; i<=n; i++){
        for(int j=1; j<=n; j++){
            if(i == j) dist[i][j] = 0;
            else dist[i][j] = INF;
        }
    }
    for(int i=0; i<m; i++){
        int u, v, d;
        cin >> u >> v >> d;
        dist[u][v] = min(dist[u][v], d);
        dist[v][u] = min(dist[v][u], d);
    }
    for(int k=1; k<=n; k++){
        for(int i=1; i<=n; i++){
            for(int j=1; j<=n; j++){
                if(dist[i][k] != INF && dist[k][j] != INF){
```

```
                dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);
            }
        }
    }
}
int min_sum = INF;
int ans = 0;
for(int i=1; i<=n; i++){
    int sum = 0;
    bool valid = true;
    for(int j=1; j<=n; j++){
        if(dist[i][j] == INF){
            valid = false;
            break;
        }
        sum += dist[i][j];
    }
    if(valid && sum < min_sum){
        min_sum = sum;
        ans = i;
    } else if(valid && sum == min_sum){
        if(ans == 0 || i < ans){
            ans = i;
        }
    }
}
cout << ans << endl;
}
signed main(){
ios::sync_with_stdio(false);
cin.tie(nullptr);
cout.tie(nullptr);
int T=1; // cin>>T;
while(T--) solve();
}
```

E 气数已尽

这道题是上学期弘深班期末考试的第四题。

主要思路是先找连通块。然后对每个连通块的周围的“气”，用 map 或者 set 进行统计，防止重复计数。

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
char grid[1005][10055];
bool visited[1005][1005];
map<pair<int, int>, int> mp;
int dx[4] = {-1, 1, 0, 0};
int dy[4] = {0, 0, -1, 1};
int n, m;
int ans1, ans2 = INT_MAX;
int cnt = 0, sz = 0;
void dfs(int x, int y) {
 if (x < 0 || x >= n || y < 0 || y >= m || grid[x][y] == '0' || visited[x][y] == true)
  return;
 }
 visited[x][y] = true;
 sz++;
 for (int i = 0; i < 4; i++) {
  int xx = x + dx[i];
  int yy = y + dy[i];
  if (xx >= 0 && xx < n && yy >= 0 && yy < m && grid[xx][yy] == '0' && mp[ {xx, yy}] ==
   cnt++;
   mp[ {xx, yy}] = 1;
  } else if (xx >= 0 && xx < n && yy >= 0 && yy < m && grid[xx][yy] == '1' && visited[x
   dfs(xx, yy);
  }
 }
}
signed main() {
 cin >> n >> m;
 for (int i = 0; i < n; i++) {
```

```
  for (int j = 0; j < m; j++) {
   char val;
   cin >> val;
   grid[i][j] = val;
  }
 }
 for (int i = 0; i < n; i++) {
  for (int j = 0; j < m; j++) {
   if (grid[i][j] == '1' && visited[i][j] == false) {
    cnt = 0;
    sz = 0;
    dfs(i, j);
    if (cnt < ans2) {
     ans2 = cnt;
     ans1 = sz;
    }
    mp.clear();
   }
  }
 }
 cout << ans1 << ' ' << ans2 << endl;
 return 0;
}
```

F 最大公因数

首先观察到以下两个性质：

- 假设其中一个数变成了 0，则另一个数可以在下一步也变成 0
- 将 a 变成 0 只需要至多 25 步

于是，最优解不超过 26。
下面给出第二个性质的证明：
考虑 a 和 b 的二进制最低位：

- 如果都是 0，则此后的 gcd 中必然包含 2 这个因子，此时将 a 和 b 除以 2 不会影响答案。

- 如果某一个数的二进制最低位不是 0 ，则 gcd 必然是一个奇数。此时将最低位不是 0 的那个数减去 gcd，最低位会变为 0。

于是，只需要至多 24 步，就能将 a 除以 4096，而 a 的最大值只有 5000。此时 a 最大只能为 1，再操作一次即可使 a 归零。

既然保证了答案不超过 26，就可以枚举每一步做了什么，使用 $O(2^26)$ 的复杂度搜索出最优解。

理论上，搜索过程中需要使用 $O(1)$ 复杂度的 gcd，但实际上很难将答案构造到 26 步。事实上，只生成出了答案为 16 的数据。所以使用 log 的 gcd 算法也可以通过。

```
//#include<bits/stdc++.h>
#include<iostream>
#include<iomanip>
#include<vector>
#include<string>
#include<cmath>
#include<algorithm>
#include<set>
#include<map>
#include<unordered_map>
#include<unordered_set>
#include<queue>
#include<stack>
using namespace std;
#define int long long
#define endl '\n'
//const int mod=1e9+7;
const int mod=998244353;
const int N=2e6+101;
int a[N];
int ans;
int gcd(int a,int b){
    if (a<b) swap(a,b);
    if (b==0) return a;
    if (a%b==0) return b;
    return gcd(b,a%b);
}
```

```
void DFS(int a,int b,int k){
    if (k>=ans) return;
    if (!a&&!b){
        ans=k;
        return;
    }
    int g=gcd(a,b);
    DFS(a-g,b,k+1);
    DFS(a,b-g,k+1);
}
void solve(){
    int a,b; cin>>a>>b;
    ans=26;
    DFS(a,b,0);
    cout<<ans<<endl;
}
signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int T=1; cin>>T;
    while (T--) solve();
}
```

G 最快的覆盖

可以先多源 BFS 求出所有点到 1 的最短距离，然后可以快速得到所有还未被覆盖的点。

我们可以自然地由二分答案钦定答案为 k，下面考虑判其是否有解：

考虑枚举所有点作为改变的点，然后等价于求其到所有未被覆盖点最大距离。可以发现，最大距离只需关注这些网格的 $\max(y+x), \min(y+x), \max(y-x), \min(y-x)$ 四个值。

时间复杂度 $O(nm\log(n+m))$。

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
#define endl '\n'
```

```
const int mod=1e9+7;
//const int mod=998244353;
const int N=2e6+101;
int n,m;
int a[N];
bool vis[N];
int dist[N];
struct node{
    int x;
    int y;
};
vector<node> vec1;
vector<node> vec2;
bool cmp1(node n1,node n2){
    return n1.x+n1.y<n2.x+n2.y;
}
bool cmp2(node n1,node n2){
    return n1.x-n1.y<n2.x-n2.y;
}
int getid(int i,int j){
    return (i-1)*m+j;
}
bool check(int t){
    for (int i=1;i<=n;i++){
        for (int j=1;j<=m;j++){
            int id=getid(i,j);
            vis[id]=0;dist[id]=1e18;
        }
    }
    queue<node> q;
    for (int i=1;i<=n;i++){
        for (int j=1;j<=m;j++){
            int id=getid(i,j);
            if (a[id]==1){
                q.push({i,j});
                dist[id]=0;
```

```
        }
    }
}
while (!q.empty()){
    node f=q.front();
    q.pop();
    int id=getid(f.x,f.y);
    if (dist[id]>t) break;
    if (vis[id]) continue;
    vis[id]=1;
    int xx=f.x,yy=f.y;
    if (xx+1<=n){
        q.push({xx+1,yy}),dist[getid(xx+1,yy)]=min(dist[getid(xx+1,yy)],dist[id]+1)
    }
    if (xx-1>0){
        q.push({xx-1,yy}),dist[getid(xx-1,yy)]=min(dist[getid(xx-1,yy)],dist[id]+1)
    }
    if (yy+1<=m){
        q.push({xx,yy+1}),dist[getid(xx,yy+1)]=min(dist[getid(xx,yy+1)],dist[id]+1)
    }
    if (yy-1>0){
        q.push({xx,yy-1}),dist[getid(xx,yy-1)]=min(dist[getid(xx,yy-1)],dist[id]+1)
    }
}
vec1.clear();
vec2.clear();
for (int i=1;i<=n;i++){
    for (int j=1;j<=m;j++){
        int id=getid(i,j);
        node nd; nd.x=i,nd.y=j;
        if (vis[id]==0) vec1.push_back(nd),vec2.push_back(nd);
    }
}
if (vec1.size()==0) return 1;
sort(vec1.begin(),vec1.end(),cmp1);
sort(vec2.begin(),vec2.end(),cmp2);
```

```
    int mi=1e18;
    for (int i=1;i<=n;i++){
        for (int j=1;j<=m;j++){
            int mx=0;
            int id=getid(i,j);
            //if (vis[id]==1) continue;
            int res;
            node nd;
            nd=vec1[0];
            res=abs(i-nd.x)+abs(j-nd.y);
            mx=max(mx,res);
            nd=vec1[vec1.size()-1];
            res=abs(i-nd.x)+abs(j-nd.y);
            mx=max(mx,res);
            nd=vec2[0];
            res=abs(i-nd.x)+abs(j-nd.y);
            mx=max(mx,res);
            nd=vec2[vec2.size()-1];
            res=abs(i-nd.x)+abs(j-nd.y);
            mx=max(mx,res);
            mi=min(mi,mx);
        }
    }
    if (mi<=t) return 1;
    return 0;
}
void solve(){
    cin>>n>>m;
    for (int i=1;i<=n;i++){
        for (int j=1;j<=m;j++){
            int id=getid(i,j);
            cin>>a[id];
        }
    }
    int l=0,r=m+n-2;
    while (l<r){
```

```
        int mid=(l+r)/2;
        if (check(mid)) r=mid;
        else l=mid+1;
    }
    cout<<l<<endl;
}
signed main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int T=1; //cin>>T;
    while (T--) solve();
}
```