A 矩形数量

```cpp
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#define N 50
using namespace std;
struct use{
    int x,y;
    bool operator < (const use &ano)const{
        return (x==ano.x ? y<ano.y : x<ano.x);
    }
  bool operator == (const use &ano)const{
    return x==ano.x && y==ano.y;
  }
}point[N];
int main(){
   // freopen("test.in","r",stdin);
    //freopen("test.out","w", stdout);

    int t, i, j, p, q, n, tot;
    scanf("%d",&t);
    while(t--){
        scanf("%d", &n);
        for (i=0;i<n;++i)
            scanf("%d%d",&point[i].x, &point[i].y);
        sort(point, point+n);
        n=unique(point,point+n)-point;
        //cout<<n<<endl;

        tot = 0;
        for (i=0;i<n;++i)
            for (j=i+1;j<n;++j){
                if (point[i].x != point[j].x)
                    break;
                if (point[i].y == point[j].y)
                    continue;
                for (p=j+1;p<n;++p){
                    if (point[p].y != point[i].y || point[p].x == point[i].x)
                        continue;
                    for (q=p+1;q<n;++q){
                        if (point[p].x != point[q].x)
                            break;
                        if (point[p].y == point[q].y || point[q].y != point[j].y)
                            continue;
                        tot += 1;
                    }
                }
            }

        printf("%d\n", tot);
    }
}
```

B 邮票收集

```cpp
#include <iostream>
```

```cpp
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
int n, m, a[105], f[1005];
int main(){
  while (scanf("%d%d", &n, &m) && (n || m)){
    for (int i = 1;i <= n;i ++) scanf("%d", &a[i]);
    f[0] = 0;
    for (int i = 1;i <= m;i ++) f[i] = 1e9;
    for (int i = 1;i <= n;i ++)
      for (int j = a[i];j <= m;j ++)
        f[j] = min(f[j], f[j - a[i]] + 1);
    printf("%d\n", (f[m] == 1e9) ? -1 : f[m]);
  }
  return 0;
}
```

C 田忌赛马

```cpp
#include <stdio.h>
#include <set>
#include <iostream>
using namespace std;
int N, X;
int enemy[12];
int ours[12];
void test(int p, int win){
    //cout<<"--"<<p<<" "<<win<<" "<<ours[0]<<" "<<ours[1]<<endl;
    int cur_win=win;
    for(int i=1;i<=N;i++){
        int used=0;
        for(int j=0;j<p;j++){
            if(ours[j]==i){
                used=1;
                break;
            }
        }
        if(used!=0)
            continue;
        ours[p]=i;
        if(i>=enemy[p]+X)
            cur_win=win+1;
        if(p==N-1){
            if(cur_win*2>N){
                for(int j=0;j<N;j++){
                    printf("%d ", ours[j]);
                }
                printf("\n");

            }
            return;
        }
        test(p+1, cur_win);
    }

}
int main(){
```

```
    cin>>N>>X;
    for(int i=0;i<N;++i){
        cin>>enemy[i];
    }
    test(0, 0);
}
```

D 漫漫回国路

```
#include<iostream>
#include<cstdio>

using namespace std;

const int MAX = 101;
int map[MAX][MAX];
bool lib[MAX]; //标记该点是否已经来过
int n, x0, y0;

bool DFS(int current)
{
    if (current == n-1) return true;
    bool flag = false;

    for(int ny = 0; ny < n; ny++){
        if (!lib[ny] && map[current][ny] > 0)
        {
            lib[ny] = 1; //走过了就不能再来
            flag = DFS(ny);
        }

        if (flag) //找到终点，返回真
            return true;
    }
    return false;
}


int main()
{
    int k;
    cin >> k;
    while (k-- > 0)
    {
        cin >> n;
        for (int i=0; i<n; i++)
        {
            lib[i] = 0;
            for (int j=0; j<n; j++)
            {
                cin >> map[i][j];
            }
        }

        if (DFS(0))
            cout << "YES" << endl;
        else
            cout << "NO" << endl;
```

```
    }

    return 0;
}

```

E Project Summer

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <iostream>
#include <vector>
#include <cassert>
#define cprintf(...) fprintf(stderr, __VA_ARGS__)
using namespace std;

int T, n, m, sx, sy, tx, ty;
char maps[105][105];
vector<pair<int,int> > gates[35];
int qx[10005],qy[10005],r,dis[105][105];

pair<int,int> other(int x,int y) {
  char p=maps[x][y];
  if('a' <=p && p<='z') {
    if(gates[p-'a'][0]==make_pair(x,y))
      return gates[p-'a'][1];
    else
      return gates[p-'a'][0];
  } else {
    return make_pair(-1, -1);
  }
}

int valid(int x,int y) {
  if(x < 0 || x >= n || y < 0 || y >=m)
    return 0;
  if(maps[x][y]=='#')
    return 0;
  return 1;
}

void push(int x,int y,int d) {
  if(valid(x,y) && dis[x][y]==-1) {
    ++r; qx[r]=x; qy[r]=y; dis[x][y]=d;
  }
}

void solve() {
  scanf("%d%d", &n, &m);
  assert(1 <= n && n <= 100);
  assert(1 <= m && m <= 100);
  for (int i = 0; i < n; ++i)
    scanf("%s", maps[i]);
  sx=sy=tx=ty=-1;
  for(int w=0;w<26;w++)
    gates[w].clear();
  for(int i=0;i<n;i++) for(int j=0;j<m;j++) dis[i][j]=-1;
```

```
  for(int i=0;i<n;i++)
    for(int j=0;j<m;j++) {
      char cur = maps[i][j];
      assert(cur == 'B' || cur == 'I' || ('a' <= cur && cur <= 'z') || cur == '.' || cur == '#');
      if(cur=='B')
        sx = i, sy = j;
      if(cur=='I')
        tx = i, ty = j;
      if('a' <= cur && cur <= 'z')
        gates[cur - 'a'].push_back(make_pair(i, j));
    }

  //cprintf("%d %d %d %d\n",sx,sy,tx,ty);
  assert(valid(sx,sy)==1);
  assert(valid(tx,ty)==1);
  for(int w=0;w<26;w++)
    assert(gates[w].size()==0||gates[w].size()==2);

  qx[r=1]=sx; qy[1]=sy; dis[sx][sy]=0;

  static int tes=0;
  for(int l=1;l<=r;l++) {
    int ux=qx[l],uy=qy[l],d=dis[ux][uy]+1;
    if(ux==tx && uy==ty) {
      printf("Case #%d: %d\n",++tes,d-1);
      return;
    }
    pair<int,int> vv = other(ux, uy);
    push(ux-1,uy,d);
    push(ux+1,uy,d);
    push(ux,uy-1,d);
    push(ux,uy+1,d);
    push(vv.first, vv.second, d);
  }
  printf("Case #%d: %d\n",++tes,-1);
  return;
}

int  main(int argc, char const *argv[])
{

  scanf("%d", &T);
  for (int i = 0; i < T; ++i)
    solve();
  return 0;
}
```

F 物资打包

```
#include <iostream>
#include <cmath>
#include <cstring>
#include <cstdio>
#include <cassert>
#include <algorithm>
using namespace std;
int n,m;
const int N=11000;
```

```
int A[N];
long long check(int now){
  long long ans=0;
  for (int i=1;i<=n;i++){
    ans+=(A[i]-1)/now+1;
  }
  return ans;
}
int main(){
  scanf("%d%d",&n,&m); assert(n>=100&&n<=1000&&m>=n&&m<=10000);
  for (int i=1;i<=n;i++){
    scanf("%d",&A[i]); assert(1000000<=A[i]&&A[i]<=10000000);
  }
  int l=1,r=0,ans=0;
  for (int i=1;i<=n;i++) r=max(r,A[i]);
  r+=1;
  while (l<r){
    int mid=l+r>>1;
    if (check(mid)<=m){
      ans=mid; r=mid;
    } else l=mid+1;
  }
  printf("%d\n",ans);
  return 0;
}
```

G 删除数字

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;


int f[1005][1005], n, a[1005];
int main(){
  int tc;
  scanf("%d", &tc);
  while (tc --){
    scanf("%d", &n);
    for (int i = 1;i <= n;i ++)
      scanf("%d", &a[i]);
    for (int i = 1;i <= n;i ++){
      for (int j = 1;j <= i;j ++){
        if (a[i] == j) f[i][j] = max(f[i - 1][j], f[i - 1][j - 1] + 1);
        else f[i][j] = max(f[i - 1][j], f[i - 1][j - 1]);
      }
    }
    int ans = 0;
    for (int i = 1;i <= n;i ++)
      ans = max(ans, f[n][i]);
    printf("%d\n", ans);
  }
  return 0;
}
```

H 最长的环

```cpp
/*
    大概思路，DFS 找出所有 0 的连通块，然后判断连通块边上的 1 是否满足构成一个环
    一些细节可能实现时候要注意一下。
    时间复杂度 O(N*M)。
*/
#include <vector>
#include <list>
#include <map>
#include <set>
#include <queue>
#include <deque>
#include <stack>
#include <bitset>
#include <algorithm>
#include <functional>
#include <numeric>
#include <utility>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cstdio>
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <cstring>
#include <cassert>

using namespace std;

typedef long long LL;
typedef pair<int, int> PII;
#define MP make_pair
#define FOR(v,p,k) for(int v=p;v<=k;++v)
#define FORD(v,p,k) for(int v=p;v>=k;--v)
#define REP(i,n) for(int i=0;i<(n);++i)
#define VAR(v,i) __typeof(i) v=(i)
#define FORE(i,c) for(__typeof(c.begin()) i=(c.begin());i!=(c).end();++i)
#define PB push_back
#define ST first
#define ND second
#define SIZE(x) (int)x.size()
#define ALL(c) c.begin(),c.end()
#define ZERO(x) memset(x,0,sizeof(x))

const int N = 1005;

int rows, cols;

char t[N][N];

bool considered[N][N];
bool visited[N][N];

int dy[] = {1, 0, -1, 0, 1, 1, -1, -1};
int dx[] = {0, 1, 0, -1, -1, 1, -1, 1};

bool isCycle(vector<PII>& v) {
```

```
58      FORE (it, v) {
59          visited[it->ST][it->ND] = false;
60          considered[it->ST][it->ND] = true;
61      }
62      int y = v[0].ST;
63      int x = v[0].ND;
64      int nVisited = 0;
65      while (y != -1) {
66          int ne = 0;
67          int nxy = -1, nxx = -1;
68          for (int d = 0; d < 4; ++d) {
69              int cy = y + dy[d];
70              int cx = x + dx[d];
71              if (considered[cy][cx]) {
72                  ++ne;
73                  if (!visited[cy][cx]) {
74                      nxy = cy;
75                      nxx = cx;
76                  }
77              }
78          }
79          if (ne != 2) {
80              break;
81          }
82          visited[y][x] = true;
83          ++nVisited;
84          y = nxy;
85          x = nxx;
86      }
87      FORE (it, v) {
88          visited[it->ST][it->ND] = false;
89          considered[it->ST][it->ND] = false;
90      }
91      return nVisited == (int) v.size();
92  }
93
94  vector<PII> boundary;
95  bool bad;
96
97  void dfs(int y, int x) {
98      visited[y][x] = true;
99      for (int d = 0; d < 8; ++d) {
100         int cy = y + dy[d];
101         int cx = x + dx[d];
102         if (t[cy][cx] == '0' && !visited[cy][cx]) {
103             dfs(cy, cx);
104         } else if (t[cy][cx] == '1' && !visited[cy][cx]) {
105             visited[cy][cx] = '1';
106             boundary.PB(MP(cy, cx));
107         } else if (t[cy][cx] != '0' && t[cy][cx] != '1') {
108             bad = true;
109         }
110     }
111 }
112
113 int main() {
114     //freopen("data.in", "r", stdin);
115     //freopen("data.out", "w", stdout);
```

```
    int T;
    cin >> T;
    while (T--) {
        ZERO(t);
        ZERO(considered);
        ZERO(visited);
        scanf("%d %d", &rows, &cols);
        for (int i = 1; i <= rows; ++i) {
            scanf("%s", t[i] + 1);
        }
        int res = 0;
        for (int i = 1; i <= rows; ++i) {
            for (int j = 1; j <= cols; ++j) {
                if (i < rows && j < cols && t[i][j] == '1' && t[i + 1][j] == '1' && t[i][j + 1] == '1' && t[i + 1][j + 1] == '1') {
                    res = max(res, 4);
                }
                if (t[i][j] == '0' && !visited[i][j]) {
                    bad = false;
                    boundary.clear();
                    dfs(i, j);
                    if (!bad && isCycle(boundary)) {
                        res = max(res, (int) boundary.size());
                    }
                    FORE (it, boundary) {
                        visited[it->ST][it->ND] = false;
                    }
                }
            }
        }
        printf("%d\n", res);
    }
}
```