A 计算鞍点

```cpp
#include <cstdio>
#include <algorithm>
#include <cstring>
#include <queue>

using namespace std;

int n=5;
int a[10][10];
bool chk(int x,int y){
  for (int i=1;i<=n;i++){
    if (a[x][i]>a[x][y]) return false;
    if (a[i][y]<a[x][y]) return false;
  }
  return true;
}
int main(){
  for (int i=1;i<=n;i++){
    for (int j=1;j<=n;j++){
      scanf("%d",&a[i][j]);
    }
  }
  for (int i=1;i<=n;i++){
    for (int j=1;j<=n;j++){
      if (chk(i,j)){
        printf("%d %d %d",i,j,a[i][j]);
        return 0;
      }
    }
  }
  printf("not found");
}
```

B 回文子串

```cpp
#include<iostream>
#include<string>
using namespace std;

string longeststring(string s)
{
        int n = s.length();

        string ans = "";
        // 枚举对称点是一个字符的所有情况
        for (int i = 0; i < n; i++) {
            // 在满足对称的情况下不断拓展长度
            int l = i, r = i;
            while (l > 0 && r < n - 1 && s[l - 1] == s[r + 1]) {
                l --; r ++;
            }
            // 与当前答案作比较
            if (r - l + 1 > ans.length()) {
                ans = s.substr(l, r - l + 1);
            }
        }
```

```
        // 枚举对称点是两个字符中间的所有情况
        for (int i = 0; i < n - 1; i++) if (s[i] == s[i + 1]) {
            // 注意初值的改变
            int l = i, r = i + 1;
            while (l > 0 && r < n - 1 && s[l - 1] == s[r + 1]) {
                l --; r ++;
            }
            if (r - l + 1 > ans.length()) {
                ans = s.substr(l, r - l + 1);
            }
        }
        // 返回答案
        return ans;
}

int main()
{
  int n;
  cin >> n;
  while( n --)  {
    string s;
    cin >> s;
    cout<<longeststring(s)<<endl;
  }
  return 0;
}

```

C 逃离迷宫

```
#include <iostream>
#include <queue>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <algorithm>

using namespace std;

int K,m,t;
char tmap[11][11];
bool visited[11][11];
int di[4] = {-1,0,0,1};
int dj[4] = {0,-1,1,0};

struct node{
  int pi;
  int pj;
  int step;
};


bool bfs(int sti, int stj, int time)
{
  node st;
  st.pi = sti;
  st.pj = stj;
  st.step = 0;
```

```
29
30    queue<node> nodes;
31    nodes.push(st);
32
33    while (!nodes.empty())
34    {
35      node cnode = nodes.front();
36      nodes.pop();
37      if (tmap[cnode.pi][cnode.pj] == 'E' && cnode.step <= time)
38      {
39        return true;
40      }
41      visited[cnode.pi][cnode.pj] = true;
42      for (int i = 0; i < 4; i++)
43      {
44        if (cnode.pi + di[i] >= 0 && cnode.pi + di[i] < m && cnode.pj + dj[i] >= 0 && cnode.pj + dj[i] < m && tmap[cnode.pi + di[i]][cnode
45        {
46          node nnode;
47          nnode.pi = cnode.pi + di[i];
48          nnode.pj = cnode.pj + dj[i];
49          nnode.step = cnode.step + 1;
50          nodes.push(nnode);
51        }
52      }
53    }
54    return false;
55  }
56
57  int main()
58  {
59    int sti, stj;
60    cin >> K;
61    while (K--)
62    {
63      cin >> m >> t;
64      memset(tmap,0,sizeof(tmap));
65      for (int i = 0; i < m; i++)
66      {
67          cin >> tmap[i];
68        for (int j = 0; j < m; j++)
69          if (tmap[i][j] == 'S')
70          {
71            sti = i;
72            stj = j;
73          }
74      }
75      memset(visited, false, sizeof(visited));
76      bfs(sti, stj, t) ? cout << "YES"<<endl :cout<< "NO" << endl;
77
78    }
79    //system("pause");
80    return 0;
81  }
```

D 拖延症

```
1   #include<bits/stdc++.h>
2
```

```cpp
using namespace std;

int w[30], s[30];

bool check(int x)
{
    bitset<32> b1(x & ((1<<5)-1));
    bitset<32> b2(x>>5);

    if(b2.count()<=b1.count()) return true;
    return false;
}

int main()
{
    int S;
    int ans, ssum, wsum;

    scanf("%d",&S);
    for(int i=0;i<15;i++)
        scanf("%d%d",&w[i],&s[i]);

    ans = 0;
    for(int i=0;i<(1<<15);i++){
        bitset<32> b(i);
        if(!check(i)) continue;
        ssum = S;
        wsum = 0;
        for(int j=0;j<15;j++){
            if(b[14-j] == 1){
                ssum-=s[j];
                wsum+=w[j];
            }
        }
        if(ssum < 0){
            continue;
        }
        if(wsum > ans) {
            ans = wsum;
        }
    }
    printf("%d\n",ans);
    return 0;
}
```

E 吃奶酪

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
const int mx = 100000;
int f[mx+10];
long long dp[mx+10];
int main()
{
  int t;
  scanf("%d",&t);
  int n;
```

```
  while( t--) {
    scanf("%d",&n);
    for(int i = 1;i <= n; ++i) {
      scanf("%d",&f[i]);
    }
    dp[0] = 0;
    if( f[1] >= 0)
      dp[1] = f[1];
    else
      dp[1] = 0;
    for(int i = 2;i <= n; ++i)
      dp[i] = max(dp[i-2] + f[i],dp[i-1]);
    cout << dp[n] << endl;
  }
  return 0;
}
```

F 图像分割

```
#include <iostream>
using namespace std;

int H, W, M;
int a[52][52];

void bfs(int i, int j)
{
  int color = a[i][j];
  a[i][j] = -1;
  if (a[i - 1][j] != -1 && abs(color - a[i - 1][j]) <= M)
    bfs(i - 1, j);
  if (a[i + 1][j] != -1 && abs(color - a[i + 1][j]) <= M)
    bfs(i + 1, j);
  if (a[i][j - 1] != -1 && abs(color - a[i][j - 1]) <= M)
    bfs(i, j - 1);
  if (a[i][j + 1] != -1 && abs(color - a[i][j + 1]) <= M)
    bfs(i, j + 1);
}

int main()
{
  while (cin >> H >> W >> M)
  {
    int count = 0;
    if (H == 0){
      return 0;
    }
    for (int i = 0; i < 52; i++){
      for (int j = 0; j < 52; j++){
        a[i][j] = -1;
      }
    }
    for (int i = 1; i <= H; i++){
      for (int j = 1; j <= W; j++){
        cin >> a[i][j];
      }
    }
    for (int i = 1; i <= H; i++){
```

```
      for (int j = 1; j <= W; j++){
        if (a[i][j] != -1){
          count++;
          bfs(i, j);
        }
      }
    }
    cout << count << endl;
  }
  return 0;
}
```

G 神奇的数列

```

#include<cstring>
#include<iostream>
using namespace std;

int a[200];
int ns[200][200];
int click_box(int start, int end) {
    int i, result, temp;
    if ( ns[start][end]>0 ) return ns[start][end];
  ns[start][end]= 1;
    if (start==end) return ns[start][end];
    result = 1 + click_box(start, end-1);
    for ( i = end - 1; i >= start; i-- ) {
        if (a[i]!=a[end]) continue;
        temp = click_box(start, i) + click_box(i+1, end-1);
        if ( temp<result ) result = temp;
    }
    ns[start][end] = result;
    return ns[start][end];
}
int main(int argc, char *argv[]){
    int t, n, i, j, m, v;
    cin >> t;
    for (i=0;i<t;i++) {
        m = 0;
    cin >> n >> a[0];
    for (j=1;j<n;j++) {
            cin >> v;
            if ( v==a[m] ) continue;
            a[++m] = v;
        }
        memset(ns,0,sizeof(ns));
        cout << "Case " << i+1 << ": " << click_box(0,  m) << endl;
    }
    return 0;
}

```

H 拯救公主

```
#include<cstdio>
#include<cstdlib>
```

```cpp
#include<deque>
using namespace std;
const int dx[4]={0,1,-1,0};
const int dy[4]={1,0,0,-1};
const int inf=1<<30;
int N,M,V;
char mp[105][105];
struct opt
{
  int x,y,v,d; //d 是方向
  opt(){}
  opt(int x,int y,int v,int d):x(x),y(y),v(v),d(d){}
  opt go(int dir){return opt(x+dx[dir],y+dy[dir],v,dir);}
  opt jump(){
    return opt(min(max(0,x+dx[d]*v),N-1),
           min(max(0,y+dy[d]*v),M-1),
           v,d);
  }
  bool valid(){return x>=0&&y>=0&&x<N&&y<M&&mp[x][y]!='#';}
  bool canjump(){return mp[x][y]=='E'&&jump().valid();}
};
struct Arr
{
  int x[105][105][11][4];  //判重标记 , 也用来记录步数
  int& operator[](const opt& X){return x[X.x][X.y][X.v][X.d]; }

  void clear(){
    for (int i=0;i<N;++i)
      for (int j=0;j<M;++j)
        for (int k=0;k<=V;++k)
          for(int d=0;d<4;++d)
            x[i][j][k][d]=inf;}
};
Arr dis;
deque<opt> q;
inline void push(opt x, int d, bool front=1)
{
  //d 是步数
  if (d<dis[x]) //如果没有重复
  {
    dis[x]=d;
    if (front)
      q.push_front(x);
    else
      q.push_back(x);
  }
}
void solve(int cas)
{
  int sx,sy,tx,ty;
  for (int i=0;i<N;++i) scanf("%s",mp[i]);
  dis.clear();
  for (int i=0;i<N;++i) for (int j=0;j<M;++j)
  {
    if (mp[i][j]=='S') sx=i,sy=j;
    if (mp[i][j]=='T') tx=i,ty=j;
  }
  opt start(sx,sy,V,0);
```

```
  dis[start]=0;
  q.push_back(start);
  while(!q.empty())
  {
    opt o=q.front();
    q.pop_front();
    for (int i=0;i<o.v;++i)
      push(opt(o.x,o.y,i,o.d), dis[o]); // throw , 放在队头
    if (o.canjump())
      push(o.jump(), dis[o]); // jump , 放在队头
    for (int i=0;i<4;++i)
      if (o.go(i).valid())
        push(o.go(i), dis[o]+1, 0); // walk 放在队尾巴
  }
  int ans=inf;
  for (int i=0;i<4;++i)
    ans=min(ans, dis[opt(tx,ty,0,i)]);
  if (ans==inf)
    ans=-1;
  printf("Case #%d: %d\n", cas, ans);
}
int main()
{

  int T=0;
  while(scanf("%d%d%d",&N,&M,&V)&&(N|M|V)) solve(++T);
  return 0;
}
```

I 宠物小精灵之对战

```
#include <stack>
#include <iostream>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <vector>
using namespace std;
char tb[504][504];
int meet[504][504];
vector<int> win[502];
int test(int st, int ed){
   if(meet[st][ed]!=-1)
       return meet[st][ed];
   meet[st][ed] = 0;
   for (int j = st + 1; j < ed; ++j)
   {
       if(test(st, j)==1&&test(j, ed)==1&&(tb[st][j]=='W'||tb[ed][j]=='W'))
           meet[st][ed] = 1;
   }
   return meet[st][ed];
}
int main()
{
   memset(meet, -1, sizeof(meet));
   int N;
   scanf("%d", &N);
   for(int i = 1; i <= N; ++i){
```

```
        scanf("%s", tb[i]+1);
        meet[i-1][i] = 1;
        meet[i][i + 1] = 1;
    }
    for (int i = 1; i <= N;++i){
        if(test(0,i)&&test(i,N+1))
            cout << i << " ";
    }
}
```

J 跳房子

```
#include"iostream"
using namespace std;

int n;
int m;
int k;
int result[30];

bool search(int dep, int n);
int compute();
int fg(int n, int i);

int compute(){
  k = 1;
  while (!search(1, n)){
    k++;
  }
  return k;
}

bool search(int dep, int n){
  if (dep>k){
    return false;
  }
  for (int i = 0; i<2; i++){
    int num = n;
    num = fg(num, i);
    result[dep] = i;
    if (num == m || search(dep + 1, num)){
      return true;
    }
  }
  return false;
}

int fg(int n, int i){
  if (i == 0)
    return 3 * n;
  else
    return n/2;
}

int main(){
  while (cin >> n >> m){
    if (n == 0){
      break;
```

```
    }
    for (int i = 0; i<30; i++){
      result[i] = 0;
    }
    k = compute();
    cout << compute() << endl;
    for (int i = 1; i <= k; i++){
      if (result[i] == 0)
        cout << "H";
      else if (result[i] == 1)
        cout << "O";
    }
    cout << endl;
  }
  return 0;
}
```

K 打炉石

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
const int inf=1<<30;
int N,K,M,H;
int a[1005],b[1005],c[1005];
int dp[2][1005][11],flag=0;
inline int min(int x,int y){return x<y?x:y;}
inline int max(int x,int y){return x>y?x:y;}
void solve()
{
  memset(dp,0,sizeof(dp));
  for (int i=0;i<N;++i)
  {
    for (int j=1;j<=M;++j) for (int k=0;k<=K;++k) dp[flag^1][j][k]=-inf;
    for (int j=1;j<=M;++j) for (int k=0;k<=K;++k) if (dp[flag][j][k]>=0)
    {
      int newhp=min(M,j+c[i])-a[i];  // heal
      if (newhp>0) dp[flag^1][newhp][k]=max(dp[flag^1][newhp][k], dp[flag][j][k]);
      if (b[i]>=H-dp[flag][j][k]) //succeed
      {
        printf("%d\n", i+1);
        return;
      }
      newhp=j-a[i];  // attack
      if (newhp>0) dp[flag^1][newhp][k]=max(dp[flag^1][newhp][k], dp[flag][j][k]+b[i]);
      if (k>0) dp[flag^1][j][k-1]=max(dp[flag^1][j][k-1], dp[flag][j][k]); // frog
    }
    flag^=1;
  }
  puts("Fail");
}
int main()
{
  scanf("%d%d%d%d",&N,&K,&M,&H);
  for (int i=0;i<N;++i) scanf("%d%d%d",a+i,b+i,c+i);
  solve();
  return 0;
```

```
}
```

L 排序

```
#include<cstdio>
#include<iostream>
#include<algorithm>
#include<cstring>
using namespace std;
#define maxn 1000
#define maxk 15
const int limits = 5;
int T,n,a[maxn],id[maxn],ans,s[maxk][maxn];
int lowerbound(int a[]) {
  //剪枝用。a[] 到排好序至少需要多少步
    int cnt = 0;
    if (a[1] != 1 ) cnt++;
    for (int i=1;i<n;i++)
        if (a[i] + 1 != a[i+1]) cnt++;  //统计有多少对乱序
    return (cnt + 2)/3; //每一步最多减少 3 对乱序
}
void trans(int a[], int b[],int p1,int p2,int p3) {
  //局面 a[] 经过一步操作变成 b[]
    for (int i=1;i<p1;i++) b[i] = a[i];
    for (int i=1;i<=p3-p2+1;i++) b[p1+i-1] = a[p2+i-1];
    for (int i=1;i<=p2-p1;i++) b[p3-p2+p1+i] = a[p1+i-1];
    for (int i = p3+1; i<=n;i++) b[i] = a[i];
}

void dfs(int dep){
  //dep 是当前深度
    if (dep + lowerbound(s[dep]) >= ans) return ;
    bool flag = 1;
    for (int i=1;i<n;i++) //看是否排好序
        if (s[dep][i]+1 != s[dep][i+1]) {
            flag = 0;
            break;
        }
    if (flag) { //已经排好序
        ans = dep; //ans = min(ans,dep) 更保险。但是由于有前面的剪枝，此处 ans 定然 <dep
        return;
    }
    for (int i=1;i<=n;i++)
        for (int j=i+1;j<=n;j++)
            for (int k=j;k<=n;k++){ //枚举所有可能操作
                trans(s[dep], s[dep+1], i, j ,k);
                dfs(dep+1);
            }
    return ;
}

bool cmp(int x, int y){
    return a[x] < a[y];
}
int main(){
    scanf("%d",&T);
    for (int _=1;_<=T;_++) {
        scanf("%d",&n);
```

```
55          for (int i=1; i<=n; i++) scanf("%d",a+i);
56          for (int i=1; i<=n; i++) id[i] = i;
57          sort(id+1,id+1+n,cmp);
58          for (int i=1; i<=n; i++) a[id[i]] = i; //此后，a[i] 表示原来的元素 i 在所有元素里面按从小到大排的序号
59      //原有元素多大没用了
60          ans = limits+1;
61          for (int i=1;i<=n;i++) s[0][i] = a[i];  //s[i][j] 表示第 i 步得到的局面
62          dfs(0);
63          printf("Case #%d: ", _);
64          if (ans > limits)  puts("Can not finish in 5 steps");
65          else printf("%d\n",ans);
66      }
67      return 0;
68  }
```