****



**程序设计综合实践课程报告**

**搜索2实验**

**学生姓名**

**学院名称 智能与计算学部**

**专 业 工科试验班(智能与计算类)**

**学 号**

# 1. 马的遍历

## 1.1题目分析

**规定好马移动的八个方向，判断好走的限制条件即可。**

## 1.2 题目代码（带注释）

|  |
| --- |
| #include <cstdio>  #include <queue>  using namespace std;  struct node  {      int x;      int y;      int step; //记录步数  };  int n, m, x, y;  int map[410][410];  int X[8] = {1, -1, 2, -2, 2, -2, 1, -1}; //8个方向  int Y[8] = {2, 2, 1, 1, -1, -1, -2, -2};  bool judge(int p, int q) //判断是否可走  {      if (p < 1 || p > n || q < 1 || q > m)          return false;      else if (map[p][q] != -1)          return false;      else          return true;  }  void BFS()  {      queue<node> q;      node t1 = {x, y, 0};      q.push(t1);      map[x][y] = 0; //不要忘了这个      while (!q.empty())      {          t1 = q.front();          q.pop();          for (int i = 0; i < 8; i++)          {              if (judge(t1.x + X[i], t1.y + Y[i]))              {                  node t2 = {t1.x + X[i], t1.y + Y[i], t1.step + 1};                  q.push(t2);                  map[t2.x][t2.y] = t2.step;              }          }      }  }  int main()  {      for (int i = 0; i < 410; i++)          for (int j = 0; j < 410; j++)              map[i][j] = -1;      scanf("%d %d %d %d", &n, &m, &x, &y);      BFS();      for (int i = 1; i <= n; i++)      {          for (int j = 1; j <= m; j++)              printf("%d ", map[i][j]);          printf("\n");      }      return 0;  } |
|  |

# 2. 求细胞数量

## 2.1题目分析

**从题意中可知细胞的定义为沿细胞数字上下左右若还是细胞数字则为同一细胞，即不论是上或下或左或右，只要不为0，则这一块是同一个细胞。简单的来说就是求连通块。**

## 2.2 题目代码（带注释）

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  int n, m;  struct xy {      int x, y;  }q[160010];  bool map[410][410], p[410][410];//p数组用来检测是否遍历过该点  char a;  int fx[4][2] = { 0,1,0,-1,1,0,-1,0 }, ans = 0;  void bfs(int x, int y)  {      ans++;      int first = 0, end = 1;      q[1].x = x;      q[1].y = y;      while (first < end)      {          first++;          int ax = q[first].x;          int ay = q[first].y;          p[ax][ay] = false;          for (int i = 0; i < 4; i++)          {              int bx = ax + fx[i][0];              int by = ay + fx[i][1];              if (map[bx][by] && p[bx][by])              {                  end++;                  q[end].x = bx;                  q[end].y = by;              }          }      }  }  int main()  {      cin >> n >> m;      memset(p, true, sizeof(p));      for (int i = 1; i <= n; i++)      {          for (int j = 1; j <= m; j++)          {              cin >> a;              if (a != '0')                  map[i][j] = true;              else                  map[i][j] = false;          }      }      for (int i = 1; i <= n; i++)      {          for (int j = 1; j <= m; j++)          {              if (map[i][j] && p[i][j])              {                  bfs(i, j);              }          }      }      cout << ans << endl;  } |
|  |

# 3. 01迷宫

## 3.1题目分析

**是一道基础的利用dfs求解的题。**

## 3.2 题目代码（带注释）

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  int n,m;  char mapp[1010][1010];//图  int dx[4]={-1,0,1,0};  int dy[4]={0,1,0,-1}; //方向数组  struct node {int x,y;};  int vis[1010][1010]={0};//标记  int anss[1010][1010]={0};//答案  void bfs(int,int);  main()  {  cin>>n>>m;  int aa,bb,t=0;  for(int i=0;i<n;i++) scanf("%s",&mapp[i]);  for(int i=0;i<m;i++)  {  cin>>aa>>bb;  vis[aa-1][bb-1]=1;   //先将自己标记  if(anss[aa-1][bb-1]) printf("%d\n",anss[aa-1][bb-1]);  else  {  bfs(aa-1,bb-1);  printf("%d\n",anss[aa-1][bb-1]);  }//如果有答案就输出否则就bfs。  }  }  void bfs(int a,int b)  {  int t=1;//答案初始为1  queue<node> q;  queue<node> qq;//联通块  node str={a,b};  node ss;  q.push(str);  qq.push(str);  while(!q.empty())  {  node now=q.front();  q.pop();  for(int i=0;i<4;i++)  {  ss.x=now.x+dx[i];  ss.y=now.y+dy[i];  if(ss.x<0||ss.x>=n||ss.y<0||ss.y>=n) continue;//越界  if(!vis[ss.x][ss.y]&&mapp[ss.x][ss.y]!=mapp[now.x][now.y])  {  t++;  vis[ss.x][ss.y]=1;  q.push(ss);  qq.push(ss);  }  }  }  while(!qq.empty())  {  node noww=qq.front();  qq.pop();  anss[noww.x][noww.y]=t;//把联通块答案记录下来  }  } |
|  |

# 4. 奇怪的电梯

## 4.1题目分析

**在深度搜索里判断电梯是往上走还是往下走即可。**

## 4.2 题目代码（带注释）

|  |
| --- |
| #include<iostream>  using namespace std;  int main()  {      int dp[205]={0};      int n,a,b;      int k[205];      int i;      bool flag=true;        cin>>n>>a>>b;      for(i=1;i<=n;i++)          cin>>k[i];    dp[a]=1;//将a层电梯初值设为1      while(flag)//至少循环一次      {          flag=false;          for(i=1;i<=n;i++)          {              if(dp[i]>0)//保证每次从a层开始              {                  if(i-k[i]>0)//保证能下降                  {                      if(dp[i-k[i]]==0||dp[i-k[i]]>dp[i]+1)                      {                          dp[i-k[i]]=dp[i]+1;//下降后次数=下降前次数+1                          flag=true;//满足条件继续循环                      }                  }                  if(i+k[i]<=n)//保证不会超出最大楼层                  {                      if(dp[i+k[i]]==0||dp[i+k[i]]>dp[i]+1)                      {                          dp[i+k[i]]=dp[i]+1;//上升后次数=上升前次数+1                          flag=true;//满足条件继续循环                      }                  }              }          }      }      cout<<dp[b]-1<<endl;        return 0;  } |
|  |

# 5. 海战

## 5.1题目分析

**见代码。**

## 5.2 题目代码（带注释）

|  |
| --- |
| #include <iostream>  #include <queue>  #include <cstdio>  #include <cstring>  using namespace std;    const int MAXN = 1000 + 10;  const int INF = 1e9;  const int dx[] = {0,0,1,0,-1};  const int dy[] = {0,-1,0,1,0};  int R, C, ans = 0;  int xx, yy, max\_x = -INF, max\_y = -INF, cnt;  char maps[MAXN][MAXN];    void dfs(int x, int y, bool t = 1)  {      if(!t)      {          ans ++;          max\_x = -INF; max\_y = -INF;          xx = x; yy = y;          cnt = 0;      }      maps[x][y] = '.';      cnt ++;      max\_x = max(max\_x, x);      max\_y = max(max\_y, y);      for(int i = 1; i <= 4; i ++)      {          int nx = x + dx[i];          int ny = y + dy[i];          if(nx >= 0 && nx <= R && ny >= 0 && ny <= C && maps[nx][ny] == '#')          {              if(nx < xx || ny < yy)              {                  ans = -1;                  return;              }              dfs(nx, ny);          }      }      if(!t && cnt < (max\_x-xx+1)\*(max\_y-yy+1))          ans = -1;  }    int main()  {      cin >> R >> C;      for(int i = 1; i <= R; i ++)          for(int j = 1; j <= C; j ++)          {              char c = getchar();              while(c != '.' && c != '#')                  c = getchar();              maps[i][j] = c;          }      for(int i = 1; i <= R; i ++)          for(int j = 1; j <= C; j ++)              if(maps[i][j] == '#' && ans != -1)                  dfs(i,j,0);      if(ans != -1)          printf("There are %d ships.", ans);      else printf("Bad placement.\n");      return 0;  } |
|  |

# 6. 路障

## 6.1题目分析

**可以利用BFS进行搜索遍历解决这个问题。**

## 6.2 题目代码（带注释）

|  |
| --- |
| #include <stdio.h>  #include <iostream>  #include <algorithm>  #include <string.h>  using namespace std;  int n,x,y,t,s,a[1001][1001],dp[1001][1001];  signed main()  {  ios::sync\_with\_stdio(false);  cin.tie(0);  cout.tie(0);  register int i,j,k;  cin>>t;  while(t--)  {  cin>>n;  memset(dp,0,sizeof dp);  memset(a,0,sizeof a);  for(i=1;i<=2\*n-2;i++)  {  cin>>x>>y;  if(x==n&&y==n&&x+y-2<i)  {  cout<<"No"<<endl;  }  if(x+y-2>i)  {  a[x][y]=1;  }  }  dp[1][1]=1;  for(i=1;i<=n;i++)  {  for(j=1;j<=n;j++)  {  if((dp[i][j-1]==1||dp[i-1][j]==1)&&a[i][j]!=1)  {  dp[i][j]=1;  }  }  }  if(dp[n][n])  {  cout<<"Yes"<<endl;  }  else  {  cout<<"No"<<endl;  }  }  return 0;  } |
|  |

# 7. 九折？

## 7.1题目分析

**贪心法的代码实现。**

## 7.2 题目代码（带注释）

|  |
| --- |
| #include<iostream>  #include<math.h>  using namespace std;    int max(int a,int b)//取最大值函数  {      return a>b ? a :b;  }    struct Thing  {      int w;      unsigned long long v;  }list[11];    unsigned long long dp[10001];    int main()  {  int Tn;  cin>>Tn;  while(Tn--){  int n;//背包容量和物品总数  long long s;  cin>>s>>n;          for(int i = 1; i <= n;i++)          {           int a;           cin>>a;           list[i].w = a\*0.9;          }          for(int i=1;i<=n;i++)          {           cin>>list[i].v;          }  for(long long i=0;i<=s;i++)dp[i]=0;//初始化二维数组          for(int i=1;i<=n;i++)//循环每个物品，逆序遍历j执行状态转移方程          {              for(long long j=s;j>=list[i].w;j--)          {                  dp[j]=max(dp[j],dp[j-list[i].w]+list[i].v);              }          }          cout<<dp[s]<<endl;      }      return 0;  } |
|  |