**算法设计与分析第二次实验**

**计科1701 2017317210122 王旭**

# 实验目的

通过此次实验掌握基本的回溯法与动态规划的思想，并且用c++语言完成几道相关题目，进一步熟悉算法的流程。

# 实验内容

实验分为两部分： **回溯法** 与 **动态规划** ，选择每个部分下两道题进行代码的编写，并测试。

## 回溯法

### 罗密欧与朱丽叶

#### 题目

问题描述：  
  
　　罗密欧与朱丽叶的迷宫。罗密欧与朱丽叶身处一个m×n 的迷宫中，如图所示。每一个方格表示迷宫中的一个房间。这m×n 个房间中有一些房间是封闭的，不允许任何人进入。在迷宫中任何位置均可沿8 个方向进入未封闭的房间。罗密欧位于迷宫的(p，q)方格中，他必须找出一条通向朱丽叶所在的(r，s)方格的路。在抵达朱丽叶之前，他必须走遍所有未封闭的房间各一次，而且要使到达朱丽叶的转弯次数为最少。每改变一次前进方向算作转弯一次。请设计一个算法帮助罗密欧找出这样一条道路。  
  
　　　　　　　　　　　　　　罗密欧与朱丽叶的迷宫  
  
编程任务：  
  
　　对于给定的罗密欧与朱丽叶的迷宫，编程计算罗密欧通向朱丽叶的所有最少转弯道路。

#### 算法思路

迷宫搜索题，直接dfs寻找所有的可能的解，然后记录每一步的方向，并比较下一步和上一步的关系，然后判断是否是题目描述的转弯情况，然后计数。除此之外，进行一定的剪枝，使算法的时空复杂度减小。

#### 实验程序

#include <bits/stdc++.h>  
using namespace std;  
const int maxn = 1e3;  
const int inf = 0x3f3f3f3f;  
  
bool mp[maxn][maxn];  
bool vis[maxn][maxn];  
int path[maxn][maxn];  
int dx[] = {0, 1, 1, 1, 0, -1, -1, -1};  
int dy[] = {1, 1, 0, -1, -1, -1, 0, 1};  
  
int ansmp[maxn][maxn];  
int ansturn, ansnum;  
int sx, sy, tx, ty;  
int n, m, k;  
int roomnum;  
  
void dfs(int x, int y, int iturn, int turn, int room, int p)  
{  
 if(x == tx && y == ty)  
 {  
 if(room != roomnum)return;  
  
 if(turn < ansturn)  
 {  
 ansnum = 1;  
 ansturn = turn;  
 for(int i = 1; i <= n; ++i)  
 for(int j = 1; j <= m; ++j)  
 ansmp[i][j] = path[i][j];  
 return;  
 }  
 if(turn == ansturn)  
 {  
 ++ansnum;  
 }  
 return;  
 }  
 if(turn > ansturn)  
 {  
 return;  
 }  
  
 for(int i = 0; i < 8; ++i)  
 {  
 int nx = x + dx[i];  
 int ny = y + dy[i];  
 if(nx >= 1 && nx <= n && ny >= 1 && ny <= m && !vis[nx][ny] && mp[nx][ny])  
 {  
 vis[nx][ny] = true;  
 path[nx][ny] = p;  
 if(x == sx && y == sy)  
 dfs(nx, ny, i, turn, room + 1, p + 1);  
 else  
 {  
 if(i != iturn)  
 dfs(nx, ny, i, turn + 1, room + 1, p + 1);  
 else  
 dfs(nx, ny, i, turn, room + 1, p + 1);  
 }  
 vis[nx][ny] = false;  
 }  
 }  
 return;  
}  
  
int main()  
{  
// freopen("input.txt", "r", stdin);  
// freopen("output.txt", "w", stdout);  
  
 while(cin >> n >> m >> k)  
 {  
 int x, y;  
 memset(mp, true, sizeof mp);  
 for(int i = 1; i <= k; ++i)  
 {  
 cin >> x >> y;  
 mp[x][y] = false;  
 }  
  
 cin >> sx >> sy >> tx >> ty;  
  
 ansturn = ansnum = inf;  
 roomnum = n \* m - k;  
 memset(vis, false, sizeof vis);  
 vis[sx][sy] = true;  
 memset(path, 0, sizeof path);  
 memset(ansmp, 0, sizeof path);  
 path[sx][sy] = 1;  
 dfs(sx, sy, 0, 0, 1, 2);  
 if(ansnum == inf)cout << "No Solution!" << endl;  
 else  
 {  
 cout << ansturn << endl << ansnum << endl;  
 for(int i = 1; i <= n; ++i)  
 {  
 for(int j = 1; j <= m; ++j)  
 if(ansmp[i][j])  
 cout << ansmp[i][j] << " ";  
 else  
 cout << -1 << " ";  
 cout << endl;  
 }  
 }  
  
 }  
  
 return 0;  
}  
  
  
  
// 3 3 1  
// 2 2  
// 1 1  
// 3 3  
// //no solution  
  
// 3 4 2  
// 1 2  
// 3 4  
// 1 1  
// 2 2  
// // 6 7 lkaf

#### 测试结果

3 3 1  
2 2  
1 1  
3 3  
//no solution  
  
3 4 2  
1 2  
3 4  
1 1  
2 2  
  
6  
7  
1 -1 9 8   
2 10 6 7   
3 4 5 -1

### 运动员最佳匹配问题

#### 题目

羽毛球队有男女运动员各n 人。给定2 个n×n 矩阵P 和Q。P[i][j]是男运动员i 和女运动员j配对组成混合双打的男运动员竞赛优势；Q[i][j]是女运动员i和男运动员j配合的女运动员竞赛优势。由于技术配合和心理状态等各种因素影响，P[i][j]不一定等于Q[j][i]。男运动员i和女运动员j配对组成混合双打的男女双方竞赛优势为P[i][j]\*Q[j][i]。  
设计一个算法，计算男女运动员最佳配对法，使各组男女双方竞赛优势的总和达到最大。  
设计一个算法，对于给定的男女运动员竞赛优势，计算男女运动员最佳配对法，使各组男女双方竞赛优势的总和达到最大。

#### 算法思路

二分图的最大权匹配问题，所以直接用KM算法就可以求解，对于两边的每一个点间的边权就是题目给的意思 就行了。

#### 实验程序

#include <bits/stdc++.h>  
using namespace std;  
const int maxn = 1e3 + 5;  
const int maxm = 2e5 + 5;  
const int inf = 0x3f3f3f3f;  
int nx, ny;  
int g[maxn][maxn];  
int linker[maxn], lx[maxn], ly[maxn];  
int slack[maxn];  
bool visx[maxn], visy[maxn];  
bool dfs(int x)  
{  
 visx[x] = true;  
 for(int y = 0; y < ny; ++y)  
 {  
 if(visy[y])continue;  
 int tmp = lx[x] + ly[y] - g[x][y];  
 if(tmp == 0)  
 {  
 visy[y] = true;  
 if(linker[y] == -1 || dfs(linker[y]))  
 {  
 linker[y] = x;  
 return true;  
 }  
 }  
 else if(slack[y] > tmp)  
 slack[y] = tmp;  
 }  
 return false;  
}  
int km()  
{  
 memset(linker, -1, sizeof linker);  
 memset(ly, 0, sizeof ly);  
 for(int i = 0; i < nx; ++i)  
 {  
 lx[i] = -inf;  
 for(int j = 0; j < ny; ++j)  
 if(g[i][j] > lx[i])  
 lx[i] = g[i][j];  
 }  
 for(int x = 0; x < nx; ++x)  
 {  
 for(int i = 0; i < ny; ++i)  
 slack[i] = inf;  
 while(true)  
 {  
 memset(visx, false, sizeof visx);  
 memset(visy, false, sizeof visy);  
 if(dfs(x))break;  
 int d = inf;  
 for(int i = 0; i < ny; ++i)  
 if(!visy[i] && d > slack[i])  
 d = slack[i];  
 for(int i = 0; i < nx; ++i)  
 if(visx[i])  
 lx[i] -= d;  
 for(int i = 0; i < ny; ++i)  
 if(visy[i])  
 ly[i] += d;  
 else  
 slack[i] -= d;  
 }  
 }  
 int res = 0;  
 for(int i = 0; i < ny; ++i)  
 if(linker[i] != -1)  
 res += g[linker[i]][i];  
 return res;  
}  
int p[maxn][maxn];  
int q[maxn][maxn];  
int main()  
{  
 int n;  
 scanf("%d", &n);  
 for(int i = 0; i < n; ++i)  
 for(int j = 0; j < n; ++j)  
 scanf("%d", &p[i][j]);  
 for(int i = 0; i < n; ++i)  
 for(int j = 0; j < n; ++j)  
 scanf("%d", &q[i][j]);  
 for(int i = 0; i < n; ++i)  
 for(int j = 0; j < n; ++j)  
 g[i][j] = p[i][j] \* q[j][i];  
  
 nx = ny = n;  
 printf("%d\n", km());  
 return 0;  
}

#### 测试结果

3  
10 2 3  
2 3 4  
3 4 5  
2 2 2  
3 5 3  
4 5 1  
Sample Output  
52

## 动态规划

### 最少硬币问题

#### 题目

　　设有n 种不同面值的硬币，各硬币的面值存于数组T［1:n ］中。现要用这些面值的硬币来找钱。可以使用的各种面值的硬币个数存于数组Coins［1:n ］中。对任意钱数0≤m≤20001，设计一个用最少硬币找钱m 的方法。

#### 算法思路

多重背包问题，直接用多重背包的思路解决即可，因为数据量不大，所以也可以转化为01背包直接求，如果不进行空间优化，可能爆时间和空间。

#### 实验程序

#include <bits/stdc++.h>  
using namespace std;  
const int maxn = 2e4 + 5;  
const int inf = 0x3f3f3f3f;  
int a[maxn], tot, n, m;  
int dp[maxn][maxn];  
int main()  
{  
 // freopen("intput.txt", "r", stdin);  
 // freopen("output.txt", "w", stdout);  
  
 ios\_base::sync\_with\_stdio(false);  
 cin.tie(0);cout.tie(0);  
 cin >> n;  
 int t, num;   
 tot = 0;  
 for(int i = 1; i <= n; ++i)  
 {  
 cin >> t >> num;  
 for(int j = 1; j <= num; ++j)  
 a[++tot] = t;  
 }  
 cin >> m;  
 if(m >= 1e5)  
 memset(dp, inf, sizeof dp);  
 else  
 for(int i = 0; i <= tot; ++i)  
 for(int j = 0; j <= m; ++j)  
 dp[i][j] = inf;  
 dp[0][0] = 0;  
 for(int i = 1; i <= tot; ++i)  
 {  
 for(int j = 0; j <= m; ++j)  
 {  
 if(j >= a[i])  
 dp[i][j] = min(dp[i - 1][j - a[i]] + 1, dp[i - 1][j]);  
 else  
 dp[i][j] = dp[i - 1][j];  
 }  
 }  
 // for(int i = 1; i <= tot; ++i)  
 // for(int j = m; j >= a[i]; --j)  
 // dp[i][j] = min(dp[i - 1][j], dp[i - 1][j - a[i]] + 1);  
  
 if(dp[tot][m] != inf)cout << dp[tot][m] << endl;  
 else cout << -1 << endl;  
 return 0;  
}

时空优化：

#include <bits/stdc++.h>  
using namespace std;  
const int maxn = 2e4 + 5;  
const int inf = 0x3f3f3f3f;  
int c[maxn], num[maxn];  
int dp[maxn];  
int n, m;  
int main()  
{  
 // freopen("input.txt", "r", stdin);  
 // freopen("output.txt", "w", stdout);  
  
 cin >> n;  
 for(int i = 1; i <= n; ++i)cin >> c[i] >> num[i];  
 cin >> m;  
 for(int i = 0; i <= m; ++i)dp[i] = inf;  
 dp[0] = 0;  
 for(int i = 1; i <= n; ++i)  
 {  
 for(int j = 1; j <= num[i]; ++j)  
 {  
 for(int k = m; k >= c[i]; --k)  
 dp[k] = min(dp[k], dp[k - c[i]] + 1);  
 }  
 }  
 if(dp[m] == inf)cout << -1 << endl;  
 else cout << dp[m] << endl;  
  
 return 0;  
}

#### 测试结果

Sample Input  
3  
1 3  
2 3  
5 3  
18  
Sample Output  
5