**实验报告**

**课程名称：算法设计与分析**

**专业班级：大数据182**

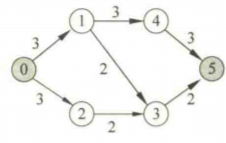
**学号姓名：3180439031陈佳婧**

**实验日期：2020年12月21日**

**第九章上机实验题3： 求解最大流最小费用问题**

1. **实验目标和要求：**





**2. 实验环境：（操作系统、语言、编译工具…）WIN10、C++、Clion**

**3. 关键问题及解决思路：**

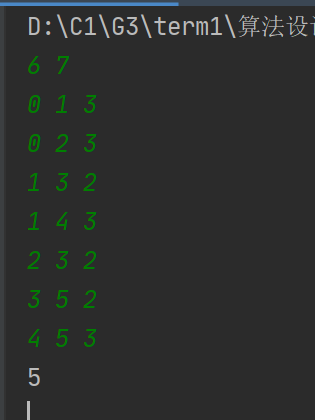
**Bellman-Ford 求最长路。**

**4. 程序流程：**

**5. 完整程序代码：**

#include<iostream>  
#include<algorithm>  
#include<cstring>  
  
using namespace std;  
const int inf = 0x3f3f3f3f;  
int n, m;  
int map[500][500];  
int vis[500];  
  
int dfs(int s, int t, int f) {  
 if (s == t)return f;  
 vis[s] = 1;  
 for (int i = 1; i <= n; i++) {  
 if (map[s][i] > 0 && !vis[i]) {  
 vis[i] = 1;  
 int d = dfs(i, t, min(f, map[s][i]));  
 if (d > 0) {  
 map[s][i] -= d;  
 map[i][s] += d;  
 return d;  
 }  
 }  
 }  
 return 0; //很重要  
}  
  
int max\_flow(int s, int t) {  
 int flow = 0;  
 while (1) {  
 memset(vis, 0, sizeof(vis));  
 int f = dfs(s, t, inf);//不断找从s到t的增广路  
 if (f == 0) {  
 return flow; //找不到了就回去  
 }  
 flow += f; //找到一个流量f的路  
 }  
}  
  
int main() {  
 while (cin >> n >> m) {  
 int u, v, w;  
 memset(map, 0, sizeof(map));  
 for (int i = 0; i < m; i++) {  
 cin >> u >> v >> w;  
 map[u+1][v+1] += w;  
 }  
 cout << max\_flow(1, n) << endl;  
 }  
}

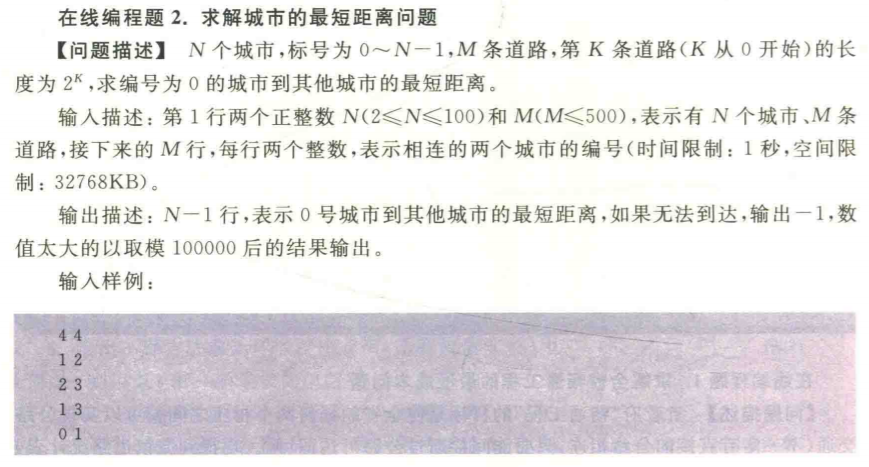
**6. 运行结果展示：**



**7. 实验体会（可选）：**

**第九章在线编程题2： 求解城市的最短距离问题**

**1.实验目标和要求：**





**2. 实验环境：（操作系统、语言、编译工具…）WIN10、C++、Clion**

**3. 关键问题及解决思路：**

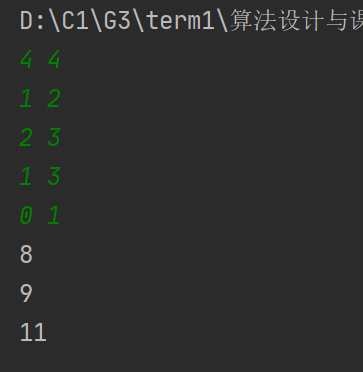
**最小生成树。**

**4. 程序流程：**

**5. 完整程序代码：**

#include <iostream>  
using namespace std;  
  
int rk[100];//记录每个树的深度  
int pre[100];//记录每个点的父节点  
int d[100][100];//记录各对间的距离  
//初始化，将每个点的父节点设为自己，深度为1  
void initSet(int n) {  
 int i;  
 for (i = 0; i < n; i++) {  
 rk[i] = 1;  
 pre[i] = i;  
 d[i][i] = 0;  
 }  
}  
  
//找到每个点的父节点，并将这个点的父节点设置为数的根节点  
int findSet(int x) {  
 if (x != pre[x])  
 pre[x] = findSet(pre[x]);  
 return pre[x];  
}  
  
//合并树，  
void unionSet(int a, int b) {  
 int x = findSet(a);  
 int y = findSet(b);  
 if (x == y)//如果两个节点的父节点（树的根节点）是同一个，无需合并，直接跳过  
 return;  
 if (rk[x] >= rk[y]) {  
 rk[x] += rk[y];  
 pre[y] = x;  
 } else {  
 rk[y] += rk[y];  
 pre[x] = y;  
 }//不是同一个树的的节点，小树合并到大树  
}  
  
//取模  
int mod(int a, int b) {  
 int ret = 1;  
 while (b--)  
 ret = (ret \* a) % 100000;  
 return ret;  
}  
  
int main() {  
 int n, m, dist;  
 int x, y, a, b;  
 int i, j, k;  
 while (cin>>n>>m&&n&&m) {  
 initSet(n);  
 for (i = 0; i < m; i++) {  
 cin>>x>>y;  
 a = findSet(x);  
 b = findSet(y);  
 if (a == b)//二者已在同一个连通分量，距离定是最小了  
 continue;  
 dist = mod(2, i);//取模  
 for (j = 0; j < n; j++)//更新两个连通分量的各对经过中间对的距离  
 {  
 if (a != findSet(j))  
 continue;  
 for (k = 0; k < n; k++) {  
 if (b != findSet(k))  
 continue;  
 d[j][k] = d[k][j] = (d[j][x] + dist + d[y][k]) % 100000;  
 }  
 }  
 unionSet(x, y);  
 }  
 x = findSet(0);  
 for (i = 1; i < n; i++)  
 if (findSet(i) != x)  
 cout<<"-1"<<endl;  
 else  
 cout<<d[0][i]<<endl;  
 }  
}

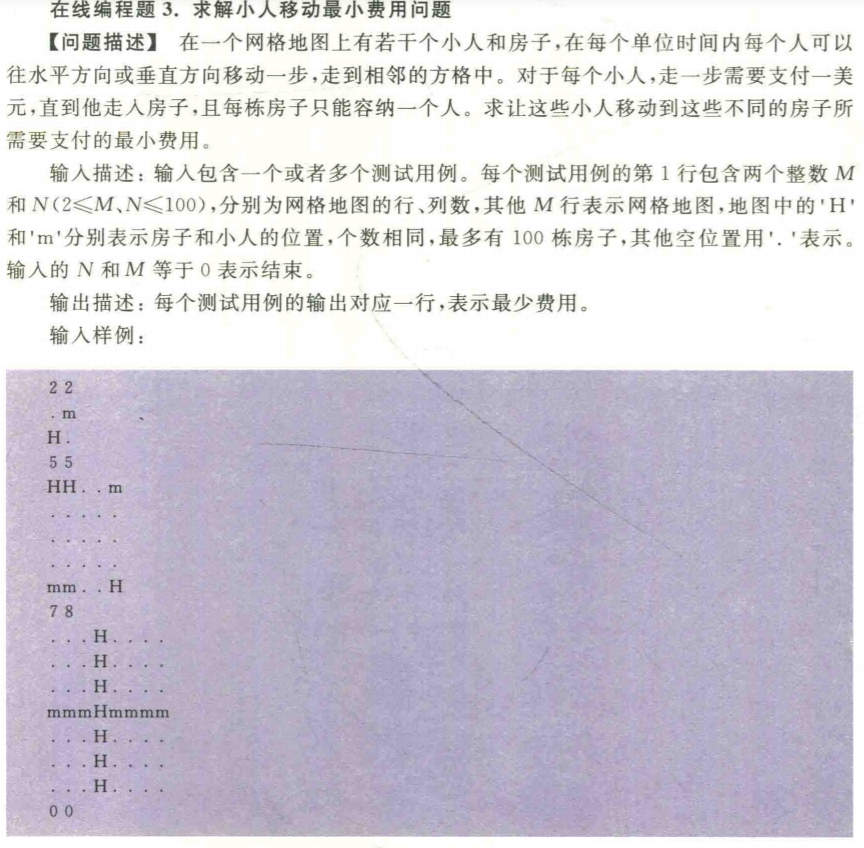
**6. 运行结果展示：**

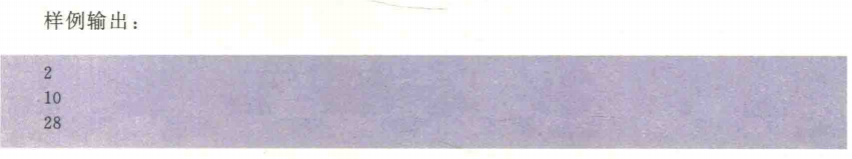


**7. 实验体会（可选）：**

**第九章在线编程题3： 求解小人移动最小费用问题**

**1.实验目标和要求：**





**2. 实验环境：（操作系统、语言、编译工具…）WIN10、C++、Clion**

**3. 关键问题及解决思路：**

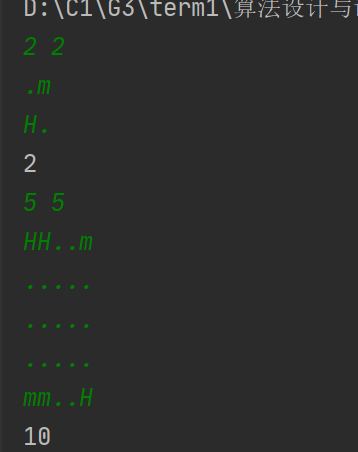
**最小费用最大流问题。**

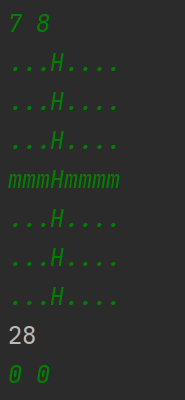
**4. 程序流程：**

**5. 完整程序代码：**

#include <iostream>  
#include <cstring>  
#include <algorithm>  
#include <queue>  
  
using namespace std;  
const int MAXN = 10000;  
const int MAXM = 100000;  
const int INF = 0x3f3f3f3f;  
struct Edge {  
 int to, next, cap, flow, cost;  
 int x, y;  
} edge[MAXM], HH[MAXN], MM[MAXN];  
int head[MAXN], tol;  
int pre[MAXN], dis[MAXN];  
bool vis[MAXN];  
int N, M;  
char map[MAXN][MAXN];  
  
void init() {  
 N = MAXN;  
 tol = 0;  
 memset(head, -1, sizeof(head));  
}  
  
//左端点，右端点，容量，花费  
void addedge(int u, int v, int cap, int cost) {  
 edge[tol].to = v;  
 edge[tol].cap = cap;  
 edge[tol].cost = cost;  
 edge[tol].flow = 0;  
 edge[tol].next = head[u];  
 head[u] = tol++;  
 edge[tol].to = u;  
 edge[tol].cap = 0;  
 edge[tol].cost = -cost;  
 edge[tol].flow = 0;  
 edge[tol].next = head[v];  
 head[v] = tol++;  
}  
  
bool spfa(int s, int t) {  
 queue<int> q;  
 for (int i = 0; i < N; i++) {  
 dis[i] = INF;  
 vis[i] = false;  
 pre[i] = -1;  
 }  
 dis[s] = 0;  
 vis[s] = true;  
 q.push(s);  
 while (!q.empty()) {  
 int u = q.front();  
 q.pop();  
 vis[u] = false;  
 for (int i = head[u]; i != -1; i = edge[i].next) {  
 int v = edge[i].to;  
 if (edge[i].cap > edge[i].flow &&  
 dis[v] > dis[u] + edge[i].cost) {  
 dis[v] = dis[u] + edge[i].cost;  
 pre[v] = i;  
 if (!vis[v]) {  
 vis[v] = true;  
 q.push(v);  
 }  
 }  
 }  
 }  
 if (pre[t] == -1) return false;  
 else return true;  
}  
  
//返回的是最大流， cost存的是最小费用  
int minCostMaxflow(int s, int t, int &cost) {  
 int flow = 0;  
 cost = 0;  
 while (spfa(s, t)) {  
 int Min = INF;  
 for (int i = pre[t]; i != -1; i = pre[edge[i ^ 1].to]) {  
 if (Min > edge[i].cap - edge[i].flow)  
 Min = edge[i].cap - edge[i].flow;  
 }  
 for (int i = pre[t]; i != -1; i = pre[edge[i ^ 1].to]) {  
 edge[i].flow += Min;  
 edge[i ^ 1].flow -= Min;  
 cost += edge[i].cost \* Min;  
 }  
 flow += Min;  
 }  
 return flow;  
}  
  
int main() {  
 int n, m;  
 while (cin>>n>>m&&n&&m) {  
 if (n == 0 && m == 0)  
 break;  
 int ch = 0, cm = 0;  
 init();//注意  
 for (int i = 0; i < n; i++) {  
 cin>>map[i];  
 for (int j = 0; j < m; j++) {  
 if (map[i][j] == 'H') {  
 HH[ch].x = i;  
 HH[ch++].y = j;  
 } else if (map[i][j] == 'm') {  
 MM[cm].x = i;  
 MM[cm++].y = j;  
 }  
 }  
 }  
 int beg = 0;//超级起点  
 int end = 2 \* ch + 1;//超级汇点  
 for (int i = 0; i < cm; i++) {  
 addedge(beg, i + 1, 1, 0);//超级起点,容量为1，花费为0  
 for (int j = 0; j < ch; j++) {  
 int tt = abs(HH[i].x - MM[j].x) + abs(HH[i].y - MM[j].y);  
 addedge(i + 1, j + 1 + ch, 1, tt);  
 }  
 addedge(i + 1 + ch, end, 1, 0);//超级汇点容量为1，花费为0  
 }  
 int ans = 0;  
 minCostMaxflow(beg, end, ans);  
 cout<<ans<<endl;  
 }  
 return 0;  
}

**6. 运行结果展示：**





**7. 实验体会（可选）：**