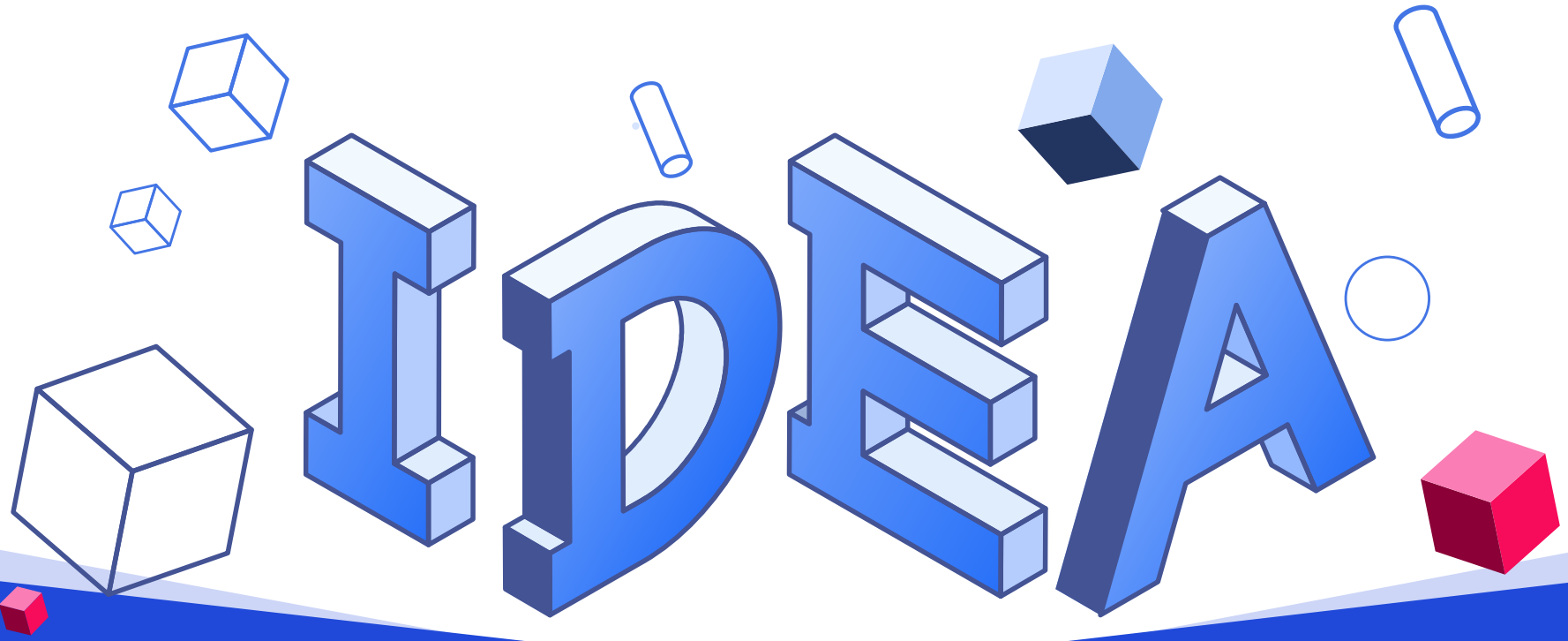


信息学

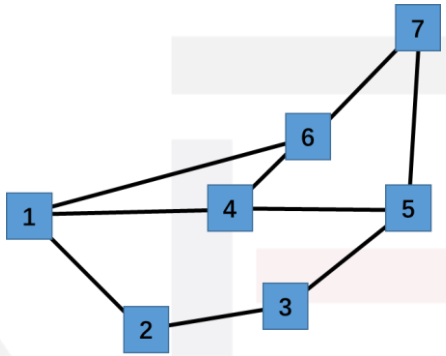
BFS习题讲解

西南大学附属中学校

信息奥赛教练组

例1：最短时间

有若干个城市，它们之间有道路连通，可以互相到达，从一个城市到另一个城市时间为1。现在给出起点城市A,终点城市B，和N条道路。问从A到B最短时间。



输入
第一行A，B，N（A,B,N<=30），B为最大城市标号

接下来N行，每行两个数x,y，表示城市x和城市y有道路相连。

输出
输出最短时间。

样例输入
1 7 9
1 2
1 4
2 3
3 5
4 5
4 6
1 6
6 7
5 7
样例输出
2

参考代码

存储起始结点信息(把起点信息放入队列)

```
while(队列不为空){
 取出队首元素u;
 标记为出队;
 将队首元素出队;
 for(遍历u的下一层未曾入队的结点){
     if(下层结点满足条件) {
        保存结点信息，结点入队
        结点标记为已入队
     }
 }

}
```

```
int a,b,n,dis[40],vis[40]={0},x,y;
bool city[40][40];
queue<int> q;
int main(){
    scanf("%d %d %d",&a,&b,&n);
    for(int i=1;i<=n;i++){
        scanf("%d %d",&x,&y);
        city[x][y]=1;
        city[y][x]=1;
    }
    q.push(a);   //队首入队
    vis[a]=1;
    dis[a]=0;
    while(!q.empty()){
        int x=q.front();  //取出队首
        q.pop();   //出队
        vis[x]=0;
        for(int i=1;i<=b;i++){
            if(city[x][i] && !vis[i]){
                q.push(i);
                vis[i]=1;
                dis[i]=dis[x]+1;
                if(i==b){
                    printf("%d",dis[i]);
                    return 0;
                }
            }
        }
    }
    return 0;
}
```

例2：求细胞个数

一矩形阵（n*m）列由数字0到9组成,数字1到9代表细胞,细胞的定义为沿细胞数字上下左右还是细胞数字则为同一细胞,求给定矩形阵列的细胞个数。

0234500067
1034560500
2045600671
0000000089
有四个细胞

输入
第一行输入n和m(n,m<20)

第二行开始为该矩阵

输出
一共有的细胞个数。

样例输入
4 10
0234500067
1034560500
2045600671
0000000089
样例输出
4

分析

第一次

```
0234500067
1034560500
2045600671
0000000089
```

```
0234500067
1034560500
2045600671
0000000089
```

第二次

```
0234500067
1034560500
2045600671
0000000089
```

```
0234500067
1034560500
2045600671
0000000089
```

第三次

```
0234500067
1034560500
2045600671
0000000089
```

```
0234500067
1034560500
2045600671
0000000089
```

第四次

```
0234500067
1034560500
2045600671
0000000089
```

```
0234500067
1034560500
2045600671
0000000089
```

多次bfs，执行了几次bfs就有几个细胞

```
for(i:1~n)
    for(j:1~m)
        if(未访问且值不为0)
            bfs()
```

分析

入队的结点信息：　　入队的是结点坐标(p,q)

之前入队的都是编号x　　q[tail++]=x;

之前入队的二维坐标(x,y)　q[tail][0]=x;　q[tail++][1]=y;

小的输入细节：一行的数字中间没有空格

采用字符串输入

参考代码(一)

常规版本

```
int main()
{
    char s[100];
    scanf("%d %d\n", &n, &m);   //这个写法可以吃掉回车
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            f[i][j] = 1;
    for (int i = 0; i < n; i++){
        fgets(s,100,stdin);
        for (int j = 0; j < m; j++){
            if (s[j] == '0')
                f[i][j] = 0;
        }
    }
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            if (f[i][j]){
               bfs(i, j);
               num++;
             }
    printf("%d\n", num);
    return 0;
}
```

```
int dx[4] = {-1,0,1,0};
int dy[4] = {0,1,0,-1};
int f[100][100], num = 0, n, m;
void bfs(int p,int q){
    int h[1000][2];
    f[p][q] = 0;
    int head=tail = 1;
    h[tail][0] = p, h[tail][1] = q;
    tail++;
    while (head<=tail){
        int x=h[head][0];
        int y=h[head][1];
        head++;
        for (int i = 0; i <= 3; i++){
            int xx = x + dx[i];
            int yy = y + dy[i];
            if ((xx >= 0) && (xx < n) && (yy >= 0) && (yy < m) && (f[xx][yy])){
                h[tail][0] = xx;
               h[tail][1] = yy;
                f[xx][yy] = 0;
                tail++;
            }
        }
    }
}
```

参考代码(二)

STL版本

```
int dx[4] = {-1,0,1,0 };
int dy[4] = {0,1,0,-1 };
int f[101][101], num = 0, n, m;
queue<pair<int,int> > Q;
void bfs(int p,int q){
    f[p][q] = 0;    //存储结点信息
    Q.push(make_pair(p,q));
    while(!Q.empty()){
        int x=Q.front().first;    //队首出队
        int y=Q.front().second;
        Q.pop();
        for (int i = 0; i <= 3; i++){    //结点搜索的状态数
            int xx = x+ dx[i];
            int yy = y+ dy[i];
            if ((xx >= 1) && (xx <= n) && (yy >= 1) && (yy <= m) && (f[xx][yy]))
            {
                Q.push(make_pair(xx,yy));
                f[xx][yy] = 0;  //入队置为0，同时也代表访问过
            }
        }
    }
}
```

```
int main() {
    char s[150];
    scanf("%d %d", &n, &m);
    getchar(); //由于下面有字符串读入，要吃掉上一行的回车
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            f[i][j] = 1;
    for (int i = 1; i <= n; i++){
        fgets(s+1, 100,stdin);    //如果要从下标为1开始存储，
gets里要写s+1
        for (int j = 1; j <= m; j++){
            if (s[j] == '0')
                f[i][j] = 0;
        }
    }
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            if (f[i][j]){  //没有访问且不为0
              bfs(i, j);
              num++;
        }
    printf("%d\n", num);
    return 0;
}
```

例3：非常可乐

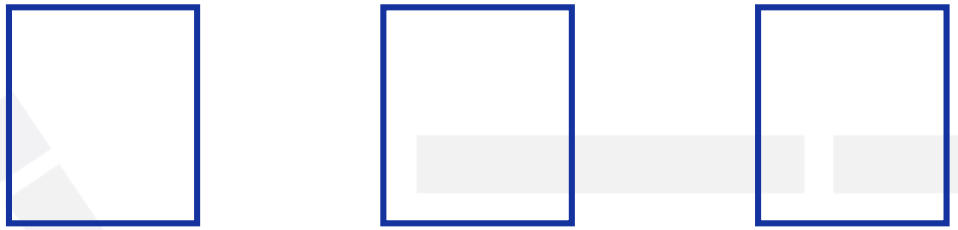
大家一定觉得运动以后喝可乐是一件很惬意的事情，但是seeyou却不这么认为。因为每次当seeyou买了可乐以后，阿牛就要求和seeyou一起分享这一瓶可乐，而且一定要喝的和seeyou一样多。但seeyou的手中只有两个杯子，它们的容量分别是N 毫升和M 毫升 可乐的体积为S （S<101）毫升 (正好装满一瓶)，它们三个之间可以相互倒可乐 (都是没有刻度的，且 S==N+M，101 > S > 0，N > 0，M > 0) 。聪明的FZOIer你们说他们能平分吗？如果能请输出倒可乐的最少的次数，如果不能输出”NO”。

输入
三个整数 : S 可乐的体积 , N 和 M是两个杯子的容量，以”0 0 0”结束。

输出
如果能平分的话请输出最少要倒的次数，否则输出”NO”。

样例输入
7 4 3
4 1 3
0 0 0

样例输出
NO
3



S(V[s]) A(V[a]) B(V[b])

V[s]=V[a]+V[b]

结点搜索六种状态

S->A S->B A->B

A->S B->A B->S

由于没有刻度，要么不倒，要么倒满

结束状态：三杯里有两杯是V[s]/2，另外一杯为0

还有一种数学推导的方法也可以解决，代码也非常简洁(线性同余方程)

参考博客：https://cloud.tencent.com/developer/article/1086892

Thanks

For Your Watching