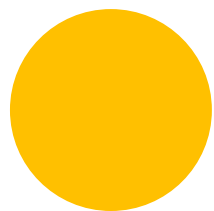
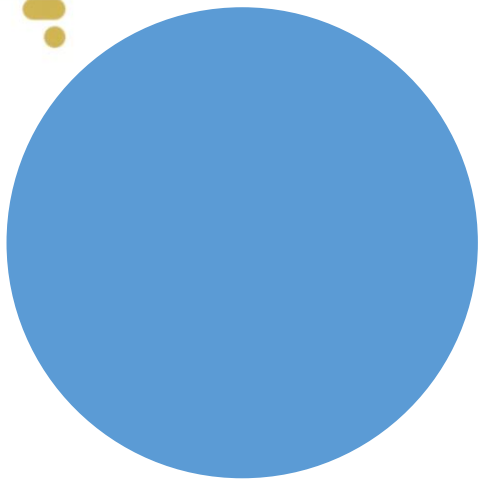


马上开始

35tang-C++竞赛系列
三阶课程





《 35tang-C++竞赛系列三阶课程 》

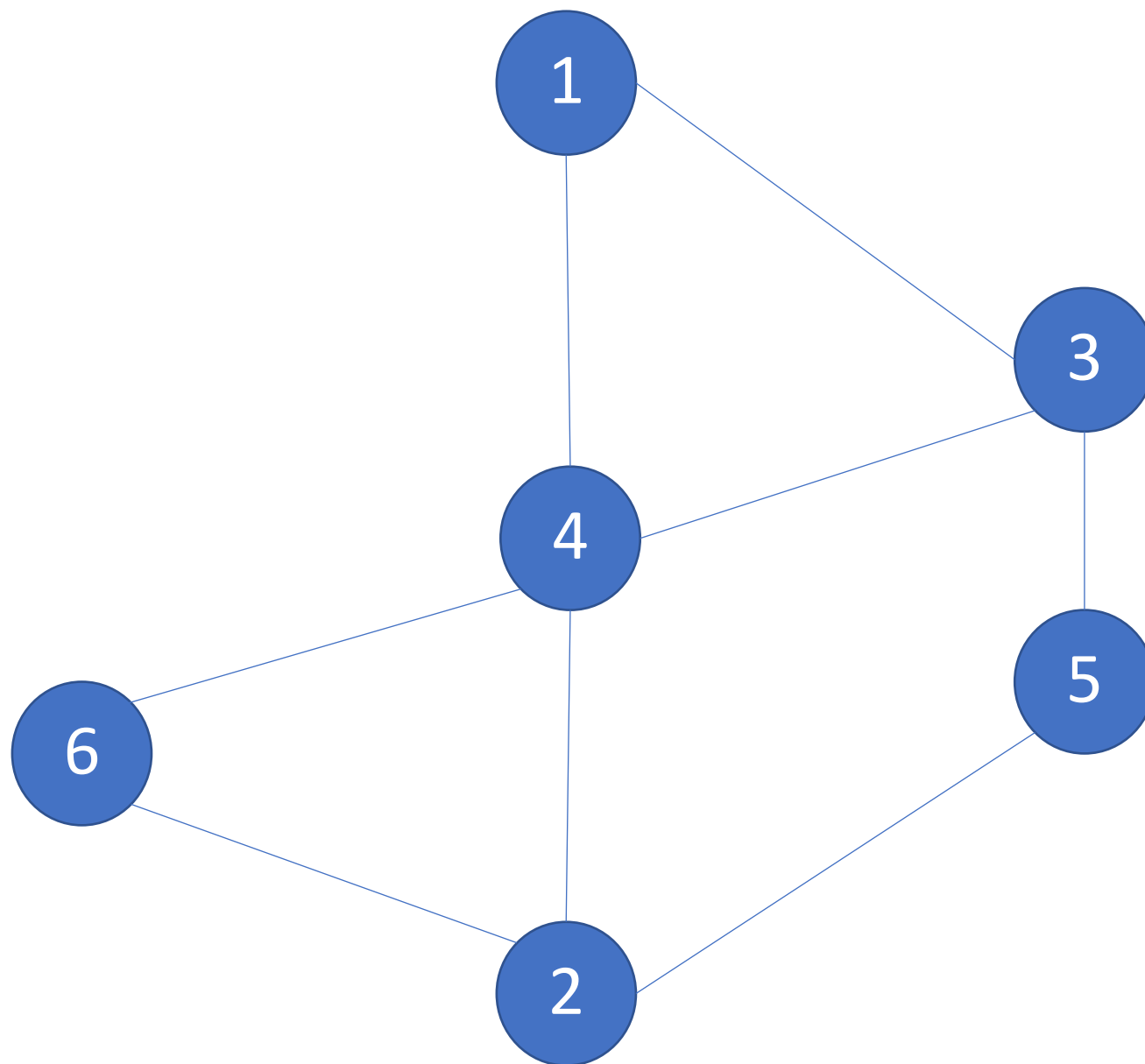


本节目标

- 进一步熟悉图;
- 掌握最短路径的搜索方法。

最短路径

- 图的基础就是搜索。DFS和BFS是基本。
- 这么一个图，如何求节点1到节点6的最短路径？假设每个点之间的权值都是1，或者说距离为1。题目实际上也是在求节点1到节点6最少经过几个点？
- 注意，右侧每一个连接是双向的，比如1 3之间有链接，表示1可以到3，3也可以到1



BFS?

节点v1 (深度为0) 入队列

While (队列不为空)

 取出队列第一个节点v

 If v是最终节点, 找到答案, 退出while循环

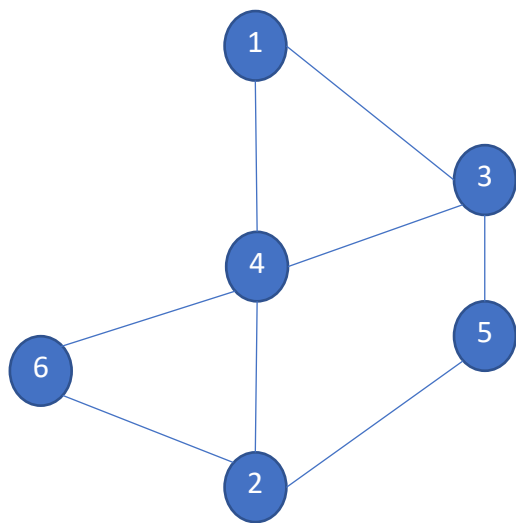
 找到v连接的每一个邻居节点, 该节点的深度=v的深度+1, 入队列

End while

因为BFS按照层次的顺序搜索, 所以退出循环时v的层次就是最短路径长度;
或者说第一次找到终节点的时候就是最短的

注意避免重复访问

- 每一个节点可能都会被再次访问，比如节点1的邻居是3和4，我们在处理3的时候会发现，1，4又是3的邻居。这就成了死循环了。
- 所以需要有一个数组来标记每一个节点是否访问过，访问过就不再访问，逻辑为什么对呢。很简单，这是BFS，一个点如果再次被访问，一定不如前面访问的路径短，所以访问过就不要再访问了。



节点v1（深度为0）入队列

`visited[节点1]=true`

While（队列不为空）

 取出队列第一个节点v

 If v是最终节点，找到答案，退出while循环

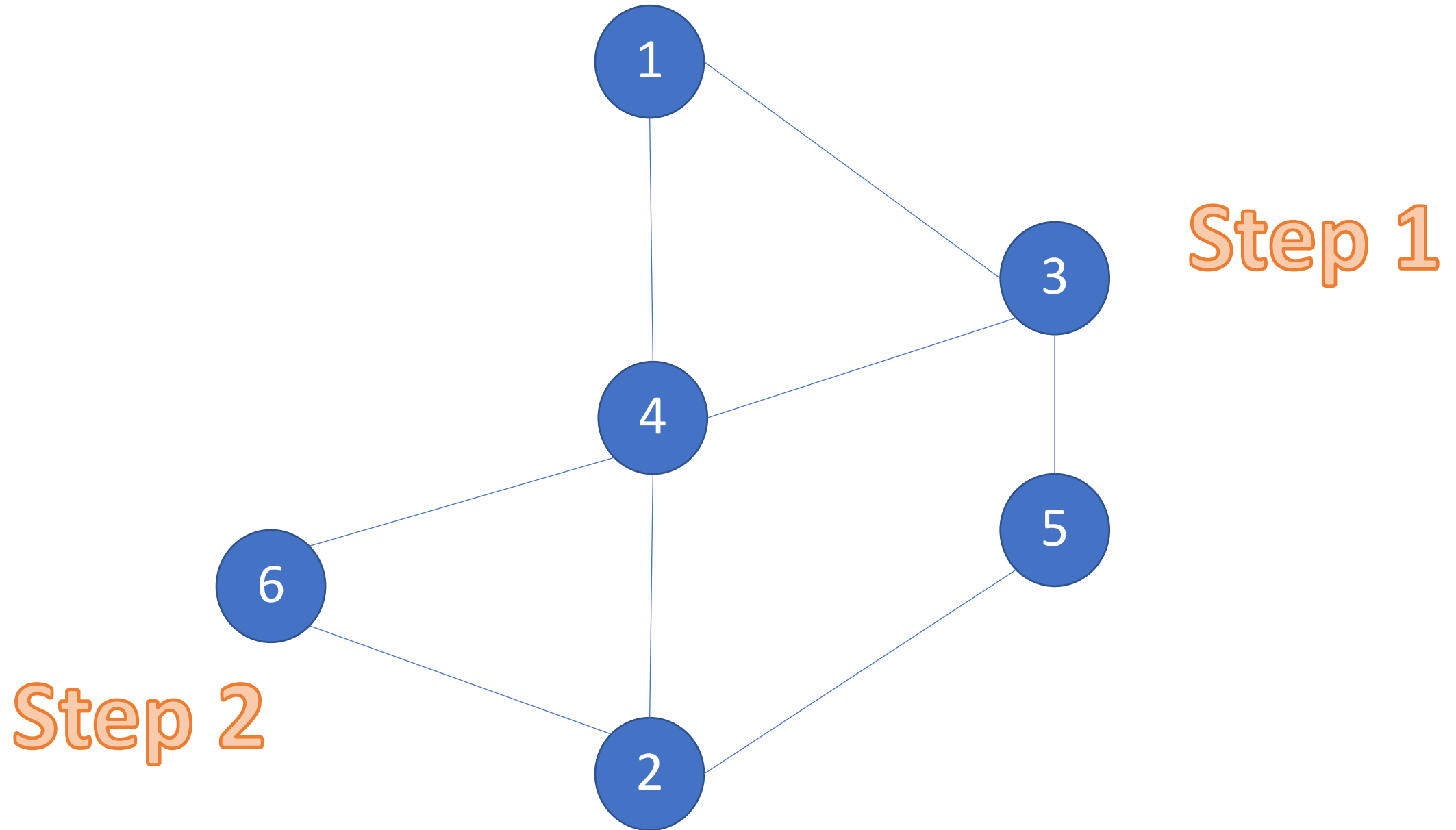
 找到v连接的每一个邻居节点vi，

 if !visited[vi] vi的深度=v的深度+1，入队列

`visited[vi]=true`

End while

退出时v的深度就是最短路径长度



迷宫问题

- 我们定义一个二维数组（矩阵结构）来表示迷宫。

```
int maze[5][5] = {  
0, 0, 0, 0, 1,  
0, 0, 0, 1, 0,  
0, 0, 0, 0, 0,  
0, 1, 1, 1, 0,  
0, 0, 0, 1, 0};
```

其中的1表示墙壁，0表示可以走的路，只能横着走或竖着走，不能斜着走，找出需要最少需要几步可以从左上角位置到达右下角出口位置。如果出不去，输出-1。



队列里面放什么？

队列里面的就是当前的一个位置，每一次队列取出来一个位置处理，就是去处理这个位置并且枚举下一步的可能（塞到队列下次处理）。所以队列面放的信息就是：知道什么才知道当前点怎么处理，下一步怎么走？

1. 当前的位置，也就是行和列，有了这个就知道下一个可能的四个方向的行列，也知道当前位置是不是到了终点
2. 当前点走了几步了，这样我们才能知道下一个位置是几步，从而判断当前是终点的时候到底走了几步。
3. 上述2个信息，行，列，步数，就是队列里面放的，用struct存放。
4. 实际上还有一个信息，就是当前访问过哪些点，从而走下一步的时候不再走，这个信息比较多，需要把所有走过的点都给记录下来。可以把他也作为队列里面的一个信息，但是太大了，所以我们把他作为全局变量visited数组，避免放到队列里。



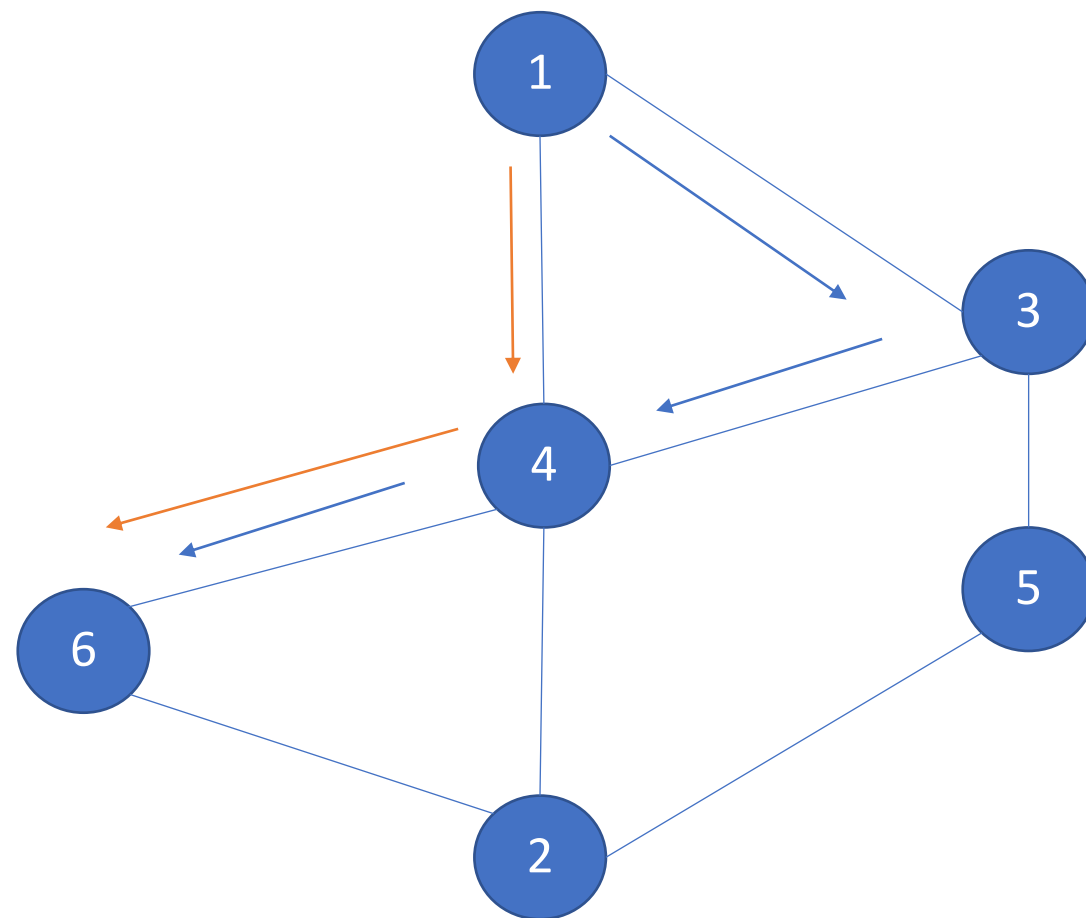
一起写



DFS

- 一样需要一个visited数组来标记是否访问过，但是这里实际上是回溯法。同样的一个节点会在不同的路径出现。

当点太多的时候，需要剪枝优化，如果当前已经走过的路线的最短长度是 a ，那么其他路线如果走到某一个位置长度已经大于 a 了，就没有必要走下去了，因为走到最后肯定大于 a 。



迷宫问题

- 我们定义一个二维数组（矩阵结构）来表示迷宫。

```
int maze[5][5] = {  
0, 0, 0, 0, 0,  
0, 0, 0, 1, 0,  
0, 0, 0, 0, 0,  
0, 1, 1, 1, 0,  
0, 0, 0, 1, 0};
```

其中的1表示墙壁，0表示可以走的路，只能横着走或竖着走，不能斜着走，找出需要最少需要几步可以从左上角位置到达右下角出口位置。如果出不去，输出-1。



dfs参数是什么

dfs就是为了选择每一条不同的道路。每一步就是处理当前位置，选择/枚举下一个位置的所有可能。他的参数也就是知道什么才知道当前点怎么处理，下一步怎么走？其实和BFS非常类似。

1. 当前的位置，也就是行和列，有了这个就知道下一个可能的四个方向的行列，也知道当前位置是不是到了终点
2. 当前点走了几步了，这样我们才能知道下一个位置是几步，从而判断当前是终点的时候这条路到底走了几步。
3. 上述2个信息，行，列，步数，就是dfs的参数。
4. 实际上还有一个信息，就是当前访问过哪些点，从而走下一步的时候不再走，这个信息比较多，需要把所有走过的点都给记录下来。其实也可以把他也作为参数传递，方便期间我们把他作为全局变量visited数组并回溯状态。



一起写

思考

- 不剪枝，出不来。为啥？想一想，每一个位置都可能有4个方向，一共多少个位置，大约24个，那么是4的24次方！！
- 就算用了剪枝，也还是比较慢。

if (steps > ans) return;

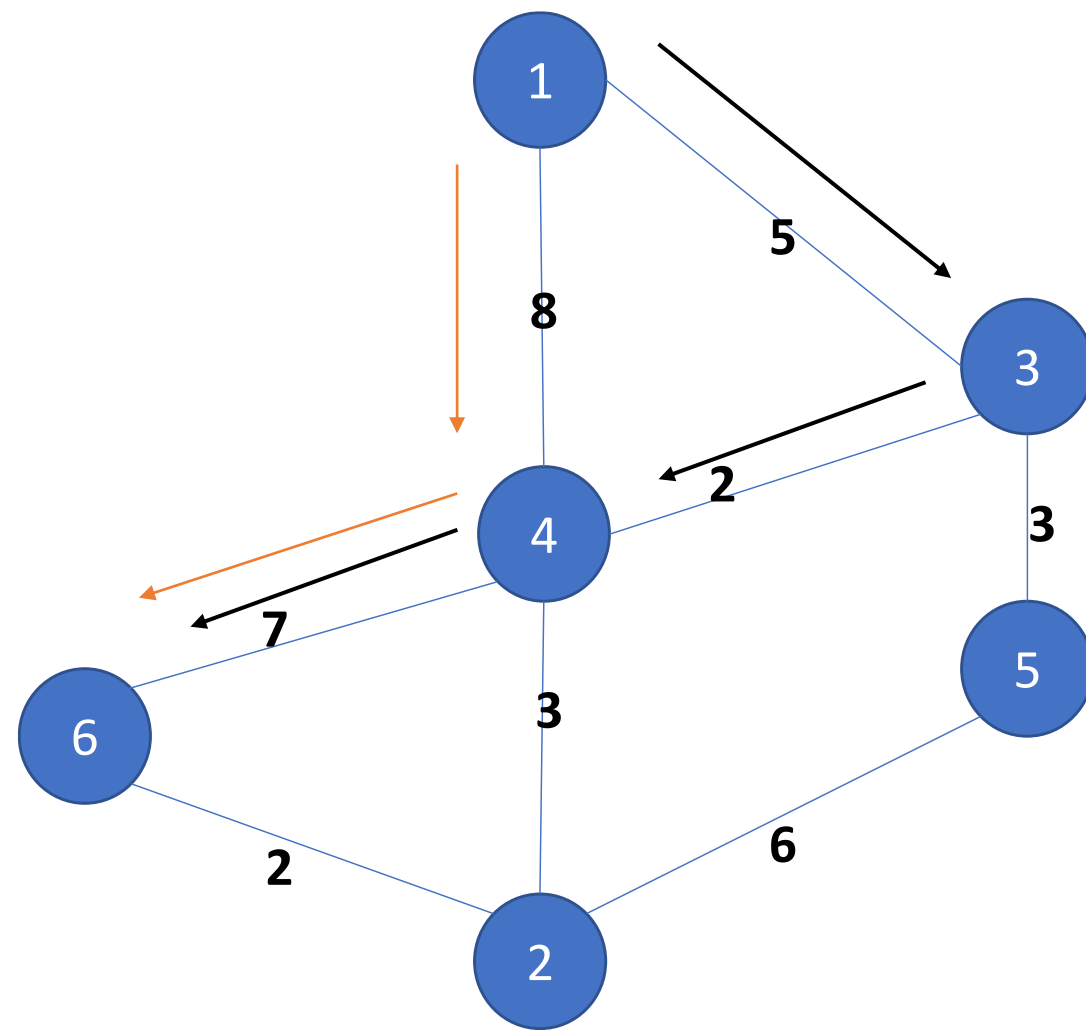
可以用记忆化剪枝或者记忆化搜索么？不可以。因为每一个位置到终点或者到起点的最短距离是不一定的，取决于你走过哪些位置，而每一个位置只能访问一次。

举个例子，位置0出发，如果一条路经过了位置1，位置2，到达位置3，这个时候最短是3。后面可能通过位置4，位置5，位置6，位置7到达终点位置8（8步）。如果另外一条路通过位置2，位置4，位置5，到了位置3，这个距离是4步，如果按照记忆化剪枝，应该不走，但是实际上位置3可能后面通过位置1，然后到达终点位置8（7步）。主要的原因，咱们第一条路搜索的时候从3往后由于1访问过了，所以不会再找到3，1，8这条最短的，而第二条路1没有走古哦，所以3之后可以找到3 1 8这条最短。

BFS也可以剪枝，但是由于每一个点只会被访问一次，所以时间复杂度是 $O(n)$ 。剪不剪意义不大。

带权值的最短路径问题

- 可以理解从站点1走到站点6，可以路过其他站点，点之间的线上的权值表示这两个站点之间的路上需要消耗的体力，比如站点1和站点3之间的路需要消耗体力5，求从站点1到站点6消耗的最小体力。
- BFS? 我们不能保证先访问到的点（近的点）就是最佳的路径的点。或者说经过点最少的路径不一定最短。
- DFS? 和前面类似的算法。



Dijkstra算法 了解就好了

- Dijkstra's algorithm determines shortest paths for one source and all destinations in $O(N^2)$ time. 注意求出的是从一个点到所有其他点的最短路径。
- 基本思路就是开始让所有点到1的距离都是无穷大，1到1的距离为0，然后循环找距离最短的没有处理过的点进行处理：找他的相邻节点，找到了就看看现在到相邻节点的路径是不是比以前发现的要短，短就更新，然后再回去找距离最短的没有处理的点。
- 本质上是贪心算法：就是每一次都会找到最近的点，去判断是否需要更新其他点是否可以通过这个新的最近点找到更近的距离。



最短路径Floyd-Warshall

求出所有点之间的最短距离

Floyd-Warshall

弗洛伊德算法

$O(n^3)$ 得出所有节点两两之间的最短距离

dist(i,j) 保存当前i到j的最佳距离，不断的找他们之间的途径点，看看有没有经过别的点可以更近的，有就更新

所有点的初始距离就是直接边的距离,如果两个点之间没有先连接呢？通过初始化设置他们为INTMAX最大

```
For i = 1 to n do
  For j = 1 to n do
    dist(i,j) = weight(i,j)
```

k是点i和点j经过的点，也就是说去找到点i通过点k然后再到点j的最小值：就是i和j之间的最短路径

```
For k = 1 to n do
  For i = 1 to n do
    For j = 1 to n do
      if (dist(i,k) + dist(k,j) < dist(i,j))
        dist(i,j) = dist(i,k) + dist(k,j)
```



USACO training Section 2.4 PROB Bessie Come Home

<https://www.luogu.com.cn/problem/P1529>

题目大意：很多奶牛用字母表示位置，互相有连接，并且给出他们之间的距离，求哪个大写字母的点到Z点的路径最短。下面示例数据表示5个节点，A和d的距离是6。

5

A d 6

B d 3

C e 9

d Z 8

e Z 3

- 示例输出：B 11

B到Z的距离是B-d-Z 11最短



分析

- 点之间的距离，就是Floyd算法中的 $\text{dist}(i,j)$ 。我们用`int pathes[52][52]`；二维数组来表示，这就是邻接矩阵表示方法，这里元素值不是1/0表示有没有连接，而是用整数来表示两个节点之间的权值（距离）。`pathes[0][1]`的值就表示节点0和节点1之间的距离。
- 第一步，把字母变为数字0到51之间的数字，作为`pathes`的数组下标，然后存入两点之间的距离。

`int chartonum(char c)//把字母映射为0到51的整数`

```
{  
    if (c>='A' && c<='Z') return c-'A';  
    else return 26+c-'a';  
}
```

经过这样的转换，`pathes[0][1]`的值就表示节点0(字母A)和节点1(字母B)之间的距离。

- 第二步就是找最短路径，由于给出的数组很小，所以完全可以用最简单的flody算法来求出所有点两两之间的距离
- 接下来我们只需要找到Z也就是到25号节点最小的

程序第一步

初始化pathes[i][j]=INT_MAX;

然后把存在直接连接的边的pathes[i][j]

修改为边的长度

5

A d 6

B d 3

C e 9

d Z 8

e Z 3

```
for (int i=0;i<52;i++)  
for (int j=0;j<52;j++)  
    pathes[i][j]= INT_MAX;
```

```
for (int k=0;k<P;k++)  
{  
    char ch1,ch2;  
    int i,j,len;  
    fin>>ch1>>ch2>>len;//读一条路，例如A d 6  
    i=chartonum(ch1);//字母变成数字  
    j=chartonum(ch2);//字母变成数字  
    if (len<pathes[i][j] )  
    {  
        //双向道路所以i和j之间以及j和i之间都有路，记下来路的长度  
        pathes[i][j]=len;  
        pathes[j][i]=len;  
    }  
}
```

第二步： Flody

```
for (int k=0;k<52;k++)  
for (int i=0;i<52;i++)  
for (int j=0;j<52;j++)  
{  
    if ((pathes[i][k]+pathes[k][j])<pathes[i][j])  
        pathes[i][j]=pathes[i][k]+pathes[k][j];  
}
```

经过flody算法， pathes[i][j]记录了i点和j点之间的最短距离。

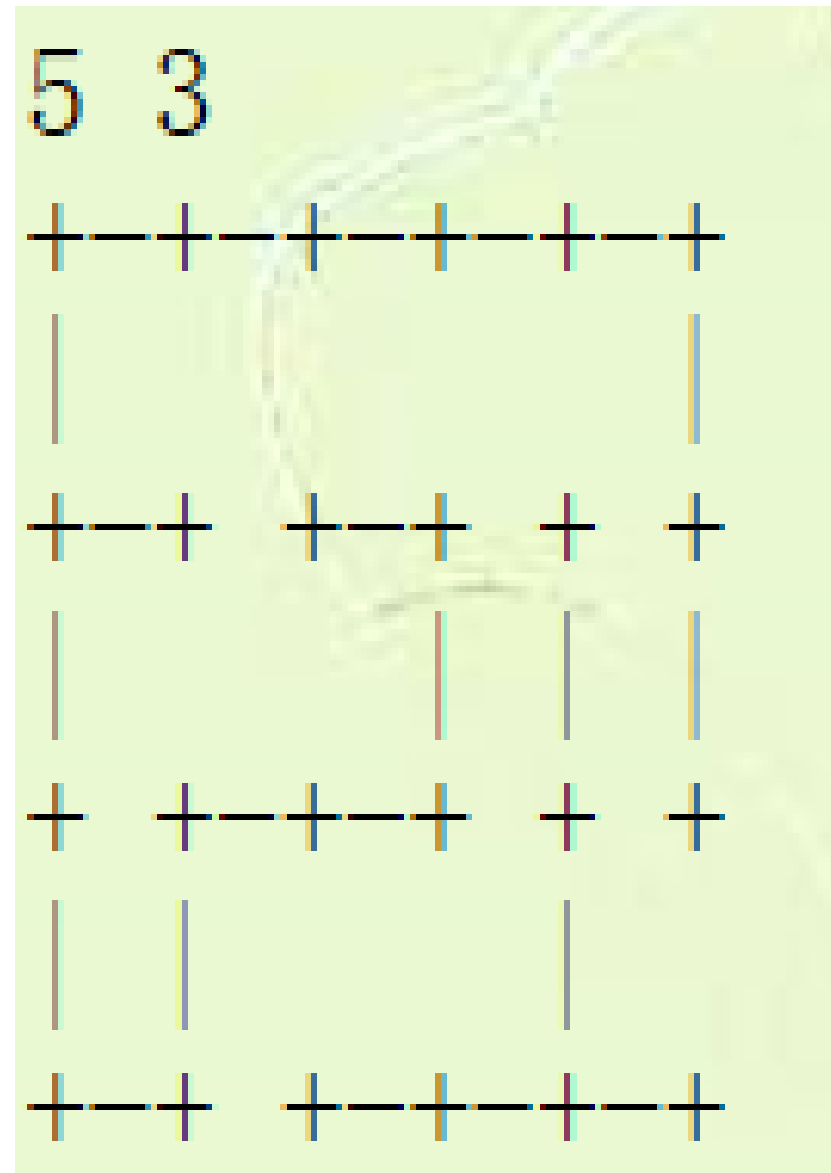
按照题目要求， 找到'Z'的距离， 'Z'按照前面转换规则转换为了点25， 那么就用i从0到24(A到Y)循环， 打擂， 找pathes[i][25]最小的那个i， 就是要找的点i， 转换为字母输出就好了

USACO training Section

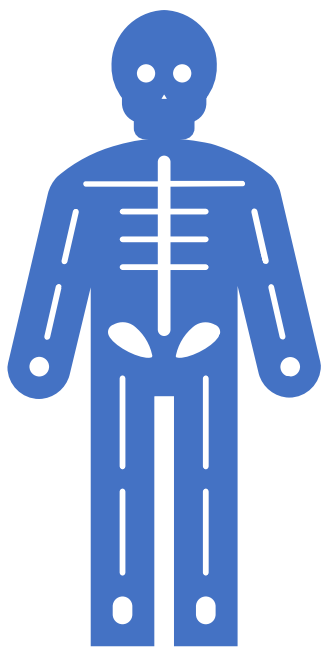
2.4 PROB Overfencing

<https://www.luogu.com.cn/problem/P1519>

- 矩阵迷宫，有两个出口。牛可能在某一个位置，找最坏位置牛走出迷宫的最小步数：也就是任意点到2个出口的最短路径的最大值。
- 换句话说：牛是聪明的，无论他现在在哪一个位置，他都会找到两个出口中路径最短的一个。现在要找的就是这些最短中最大的一个。
- 这里的+ - 等表示墙壁



算法



- DFS, Dijkstra's或者Floyd-Warshall都可以, 由于这道题要求出所有点到出口的最短路径而且数据量不大。用Floyd-Warshall最简单, 不过由于题目直接读取的是一个矩阵, 所以我们还需要先需要建立边的连接关系, 把矩阵里面的一个位置(行和列)转换为一个整数表示的节点编号。因为邻接矩阵是二维的, 每一个维度是一个节点的编号, 而不是行列两个整数。
- 还有一个方法就是BFS, 因为这个相当于权值为1的图, 而且每一个节点的相邻节点最多4个。BFS找最短就是去找经过几个位置到达出口。可以对矩阵中每一个位置都走一次BFS寻找他和两个出口的最短距离(两个出口, 先到的最短), 对所有位置找到的这样的最短距离, 打擂台找其中的最大。



```
for (int row=1;row<2*H;row+=2)
```

```
for (int col=1;col<2*W;col+=2)
```

```
//对每一个位置都要用一遍BFS寻找他和出口的距离
```

```
{
```

```
    operationpoints=queue <ps> (); //一定要清空， 因为前面会break出来
```

```
    bool visitedflag[201][77]={0};
```

```
    ps currentpoint={row,col,0};
```

```
    operationpoints.push(currentpoint);
```

```
    visitedflag[currentpoint.row][currentpoint.col]=true;
```

```
    while (!operationpoints.empty()) {
```

```
        ps oldpoint=operationpoints.front();
```

```
        operationpoints.pop();
```

```
        if (isexist(oldpoint.row,oldpoint.col)) { //找到出口， 也就是一个最短路
```

```
            if (oldpoint.len+1>r) r=oldpoint.len+1; //打擂台找最短之中的最大
```

```
            break;
```

```
        }
```

```
        if (grid[oldpoint.row-1][oldpoint.col]!='-'&&!visitedflag[oldpoint.row-2][oldpoint.col]) {
```

```
            ps newpoint={oldpoint.row-2,oldpoint.col,oldpoint.len+1};
```

```
            operationpoints.push(newpoint);
```

```
            visitedflag[oldpoint.row-2][oldpoint.col]=true;
```

```
        }
```



如果矩阵很大，怎么优化？

- 就是反过来：不是从每一个位置启动BFS去找到出口的最短，而是从两个出口去找出口到每一个位置的最短距离。具体算法：把两个出口位置入队列然后开始BFS，对于其他的位置，如果通过BFS按照相邻节点的方式找到了，第一次找到的步数就是最短路径，因为两个出口是同一个层次。所有找到的最短去打擂台找最大。
- 需要避免重复访问一个节点，可以用二维数组存放每一个位置到两个出口任意一个的最小值，找到了后面就不需要再找了。也可以用visited数组来表示访问状态。
- 当我们找到一个位置的时候，无论是从哪一个出口找过来的，由于两个出口在同一个层次，最先找到这个位置时候的步数一定是这个位置到两个出口的最短路径

总结

如果没有权值，用BFS是最简单的。

如果有权值，BFS不好做，可行的方法：

- DFS回溯，搜索所有路径，打擂台最短，效率很低，需要剪枝
- Dijkstra's, $O(n^2)$
- Floyd-Warshall, $O(n^3)$, 所以数据量不能太大了

下节课预习

- 后面我们要进入冲刺阶段了。题目都会比较难，基本上是NOIP第三四题的水平。
- 冲刺第一节讲二分，大家可以回忆一下二分法。



作业1 angel.cpp

拯救天使，一个迷宫中，a表示天使的位置，有一些守卫用x表示，r表示救援队伍。现在救援队伍需要到达a的位置从而拯救天使。

假设移动需要1单位时间，杀死一个看守也需要1单位时间。到达一个格子以后，如果该格子有看守，则一定要杀死。交给你的任务是，最少要多少单位时间，才能到达天使所在的地方？
(只能向上、下、左、右4个方向移动)

输入：第一行二个整数n，m。表示迷宫的大小为n*m (N, M <= 200)。以后n行，每行m个字符。其中“#”代表墙，“.”表示可以移动，“x”表示看守，“a”表示天使，“r”表示救援队伍。字母均为小写。

输出：一行，代表救出天使的最短时间。如果救援小组永远不能达到天使处，则输出“NO ANSWER”

提示：无权值，DFS，BFS都可以,DFS一定要剪枝

样例输入

7 8

#####.

#.a#..r.

#..#x...

..#..#.#


#...##..

.#.....

.....

样例输出

13



挑战作业2：奶牛吃草 grass.cpp

n 个编号为1到 n 的牛棚，每两个牛棚之间的路需要消耗奶牛的体力值是一个大于0的整数，奶牛从1号牛棚出发，要求所有牛棚经过并且只经过一次。选择一条路，让奶牛可以消耗最小的体力。输出这个最小体力值。

注意：请自行根据示例数据构造输入文件，并在本机测试通过，要求执行时间在1s之内。

输入格式（文件名stall.in）：

第一行一个整数 n 表示 n 个牛棚。 $n \leq 12$ 。

下面 n 行 n 列的矩阵，表示牛棚之间的消耗体力值，体力值不会超过10000。矩阵的第 i 行第 j 列的数字表示牛棚 i 到牛棚 j 消耗的体力。注意 i 牛棚到 j 牛棚的路和 j 牛棚到 i 牛棚可能不是一条路，消耗体力不同。第 i 行第 j 列的数字为0表示 i 到 j 没有路。

输出（文件名stall.out）：

1个整数。表示走完所有牛棚消耗体力最小值。

挑战作业2 说明



示例输入1:

4

0 1 2 3

2 0 3 4

5 4 0 7

1 2 3 0

示例输出2:

8

上面表示3号牛棚到4号牛棚消耗体力7，4号到3号消耗体力3，从1号到4号消耗体力3，4号到2号消耗体力2，2号到3号消耗体力3...这样从牛棚1到牛棚4再到牛棚2再到牛棚3，走完4个牛棚，总体力消耗这 $3+2+3=8$ 是最小的一种方案。

提示：这个矩阵和迷宫不同，不是用来表示障碍或者相邻位置，而是表示体力值。每一个牛棚的相邻位置其实有 $n-1$ 个，也就是说某一个牛棚可以走到其他 $n-1$ 个牛棚的任意一个，而消耗的体力值从这个矩阵查找。

提示：用DFS搜索简单一些，从节点1开始，DFS他的所有相邻节点，注意回溯。DFS搜索的时候需要传递的参数除了当前行列，还需要当前路线消耗的体力和，以及当前经过了几个节点。当我们找到 n 个点的时候就是找到了一条路，这时候的体力和就是这条路线的消耗总体力，多条路线打擂台法找这个的最小。

注意：一定要剪枝优化，否则下一个示例数据爆掉

示例输入2:

12

0 1 2 3 4 5 6 7 8 9 12 13

2 0 3 4 7 8 9 11 12 13 4 5

12 1 0 3 4 5 6 7 8 9 12 13

2 0 3 0 7 8 9 11 12 13 12 13

4 1 2 3 0 5 6 7 8 9 12 13

2 13 3 4 7 0 9 11 12 13 4 1

10 1 2 3 4 5 0 7 8 9 4 1

2 11 3 4 7 8 9 0 12 13 4 1

8 1 2 3 4 5 6 7 0 9 4 1

2 6 3 4 7 8 9 11 12 0 7 8

2 6 3 4 7 8 9 11 12 13 0 8

4 1 2 3 10 5 6 7 8 9 12 0

示例输出2:

51

挑战作业2 示例数据2

下载“课堂讲义” pdf文件
到本地，然后复制数据
到输入文件进行测试

3 超级挑战作业 tours.cpp

- USACO training Section 2.4 PROB Cow Tours
- 洛谷题目编号: P1522
- 给出一些坐标点, 通过矩阵表示这些点是否互相连接, 联通其中两个点, 使得更新后任意连通的两点距离最小。



作业3提示

- 重点在于搞清楚更新后任意两点距离最小什么意思。算法不难，先求出所有点的最小距离，然后找到每一个点到其他所有连接点的最大距离，然后枚举所有不连通的点，计算联通后的最大最短路径（就是链接的两个点的距离+这两个点分别到其他所有点的最短路径的最大值），和原来的最大比较。
- 找最短路径，数据量不大，显然用floyd算法简单一些。



由易到难，思维体系训练
实战结合，创新协作培养
兴趣导向，未来职业引领

<https://www.35tang.com>



扫码关注公众号

<https://www.三五堂.com>



添加辅导老师