声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

可达信奥_小武老师_keda.ac

《L4-CSP-J 复赛进阶算法与数据结构》

Day04 广度优先搜索BFS优化技巧

主讲人：小武老师

可达信奥_小武老师_keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

可达信奥_小武老师_keda.ac

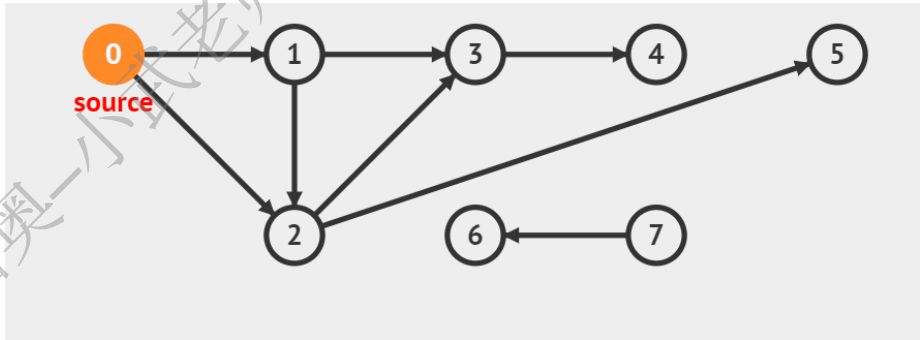
广度优先搜索

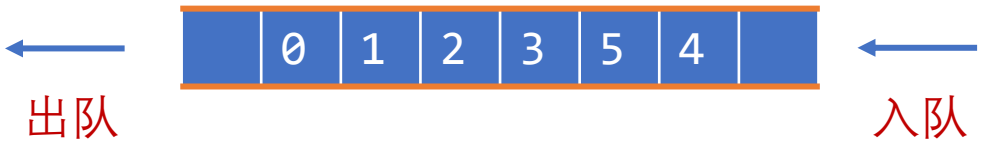
BFS 本质就是让你在一幅「图」中找到从起点 start 到终点 target 的最近距离。对的，就是这么朴实无华且枯燥。BFS优化的本质是减少搜索次数或加快搜索效率。

可达信奥_小武老师_keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

广度优先搜索





BFS 的核心数据结构是队列

```
while (队列不为空) {
    int u = 队首;
    弹出队首;
    for (枚举 u 的邻居) {
        更新数据
        if (...)
            添加到队首;
        else
            添加到队尾;
    }
}
```

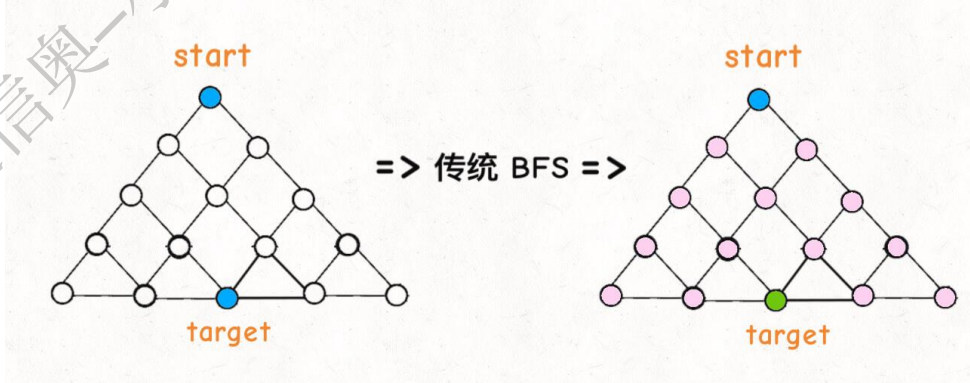
BFS 的核心框架

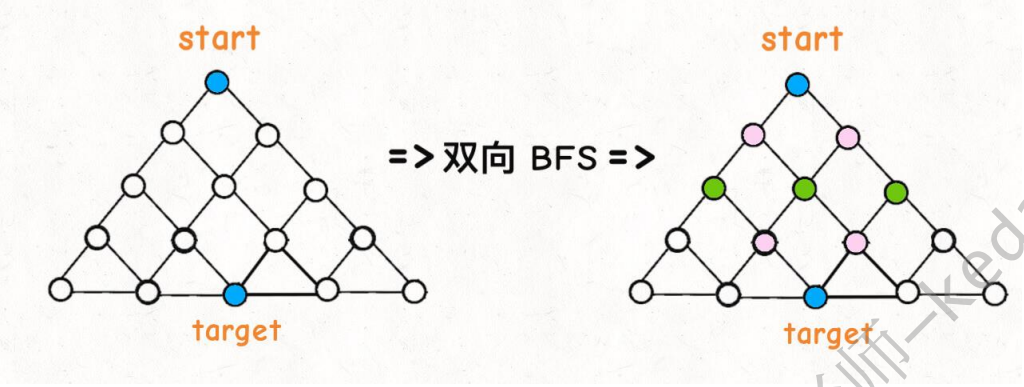
可达信奥_小武老师_keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

双向BFS

传统的 BFS 框架就是从起点开始向四周扩散，遇到终点时停止；而双向 BFS 则是从起点和终点同时开始扩散，当两边有交集的时候停止。





而双向 BFS 其实只遍历了半棵树就出现了交集，也就是找到了最短距离

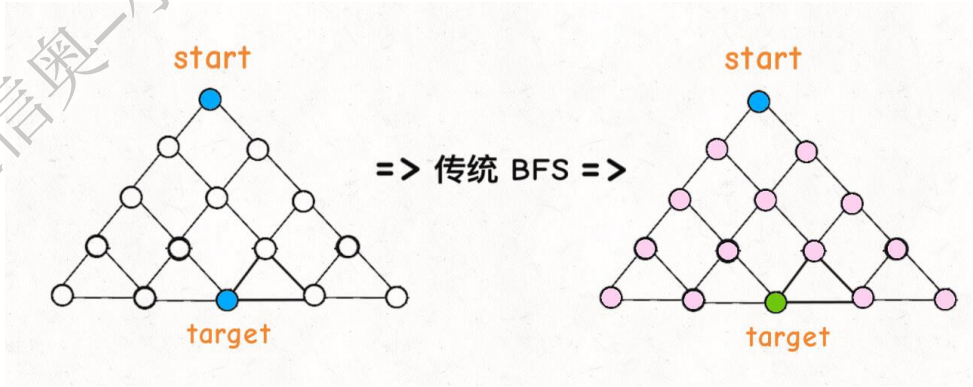
双向 BFS 也有局限，你必须知道终点在哪里

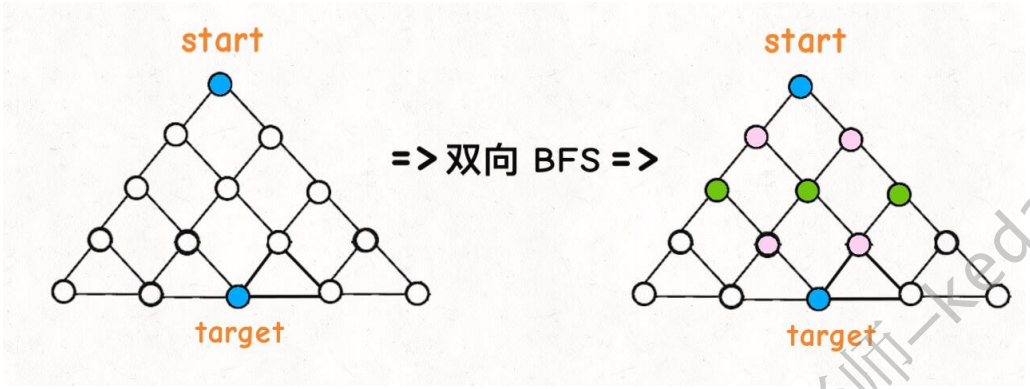
可达信奥-小武老师-keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

双向BFS

传统的 BFS 框架就是从起点开始向四周扩散，遇到终点时停止；而双向 BFS 则是从起点和终点同时开始扩散，当两边有交集的时候停止。





无论传统 BFS 还是双向 BFS，无论做不做优化，从 Big O 衡量标准来看，时间复杂度都是一样的，只能说双向 BFS 是一种 trick，算法运行的速度会相对快一点，掌握不掌握其实都无所谓。最关键的是把 BFS 通用框架记下来，反正所有 BFS 算法都可以用它套出解法。

可达信奥-小武老师-keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

可达信奥_小武老师_keda.ac

编程实践 Online Judge

P0094 最小步数

P0420 走迷宫 优化

P0421. 抓住那头牛(单向+双向优化)

P0436. 八数码难题

可达信奥_小武老师_keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

双向BFS

P0420 走迷宫 （双向队列优化）

```
void bfs(int sx, int sy, int ex, int ey){
    int head = 1, tail = 1;
    int head2 = 1, tail2 = 1;
    dis[sx][sy] = 1, dis[ex][ey] = 1;
    q[tail].x = sx;   q2[tail2].x = ex;
    q[tail].y = sy;   q2[tail2].y = ey;
    tail++; tail2++;
    vis[sx][sy] = 1;  vis[ex][ey] = 2;
    bool flag;
    while(head < tail && head2 < tail2){
        int qsize1 = tail - head;
        int qsize2 = tail2 - head2;
        int x0,y0,step;
        if(qsize1 > qsize2){
            x0 = q2[head2].x;
            y0 = q2[head2].y;
            flag = 0;
            head2++;
        }else{
            x0 = q[head].x;
            y0 = q[head].y;
            flag = 1;
            head++;
        }
```

```
for(int i = 0; i < 4; i++){
            int nx = x0 + d[i][0];
            int ny = y0 + d[i][1];
            if(nx >= 0 && nx < n && ny >= 0  && ny < m && a[nx][ny] !=
'#'){ //vis[nx][ny] != 1
                if(!vis[nx][ny]){
                    if(flag){
                        q[tail].x = nx;
                        q[tail].y = ny;
                        dis[nx][ny] = dis[x0][y0] + 1;
                        tail++;
                        vis[nx][ny] = 1;
                    }else{
                        q2[tail2].x = nx;
                        q2[tail2].y = ny;
                        dis[nx][ny] = dis[x0][y0] + 1;
                        tail2++;
                        vis[nx][ny] = 2;
                    }
                }else{
                    if(vis[nx][ny] + vis[x0][y0] == 3){
                        cout << dis[nx][ny] + dis[x0][y0];
                        return;
                    }
                }
            }
        }
    }
}
```

可达信奥_小武老师-keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

P0421. 抓住那头牛（单向BFS）

```
void bfs(int sx, int ex){
    int head = 1, tail = 1;
    q[tail].x = sx; q[tail].step = 0;
    tail++;
    vis[sx] = 1;
    while(head < tail){
        int x0 = q[head].x;
        int step = q[head].step;
        if(x0 == ex){
            cout << step << endl;
            return;
        }
```

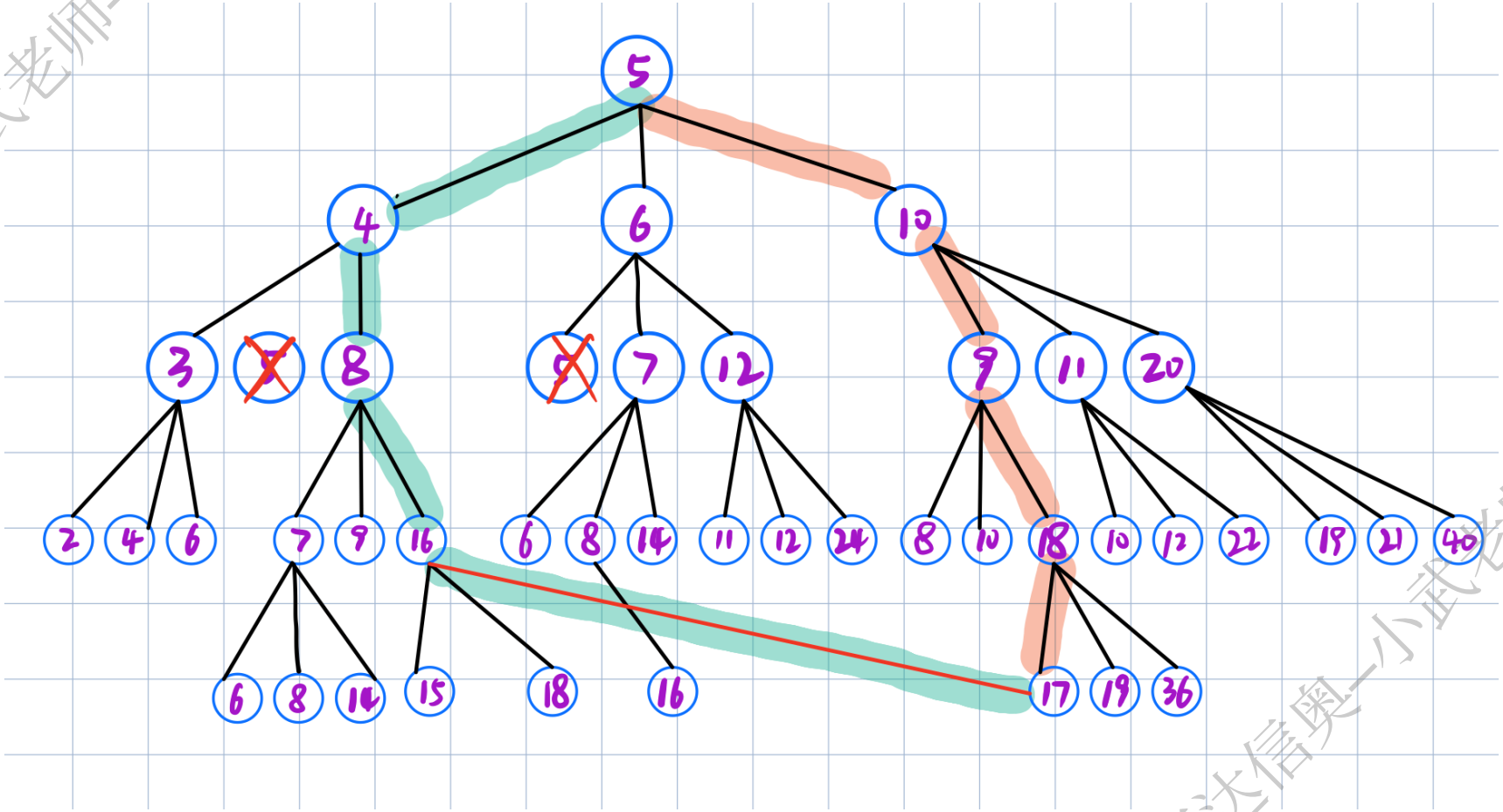
```
 for(int i = 1; i <= 3; i++){
            int nx;
            if(i==1) nx = x0 + 1;
            if(i==2) nx = x0 - 1;
            if(i==3) nx = x0 * 2;
            if(nx >= 0 && nx <= 100000 && !vis[nx]){
                q[tail].x = nx;
                q[tail].step = step + 1;
                tail++;
                vis[nx] = 1;
            }
        }
        head++;
    }
}
```

可达信奥-小武老师-keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

双向BFS

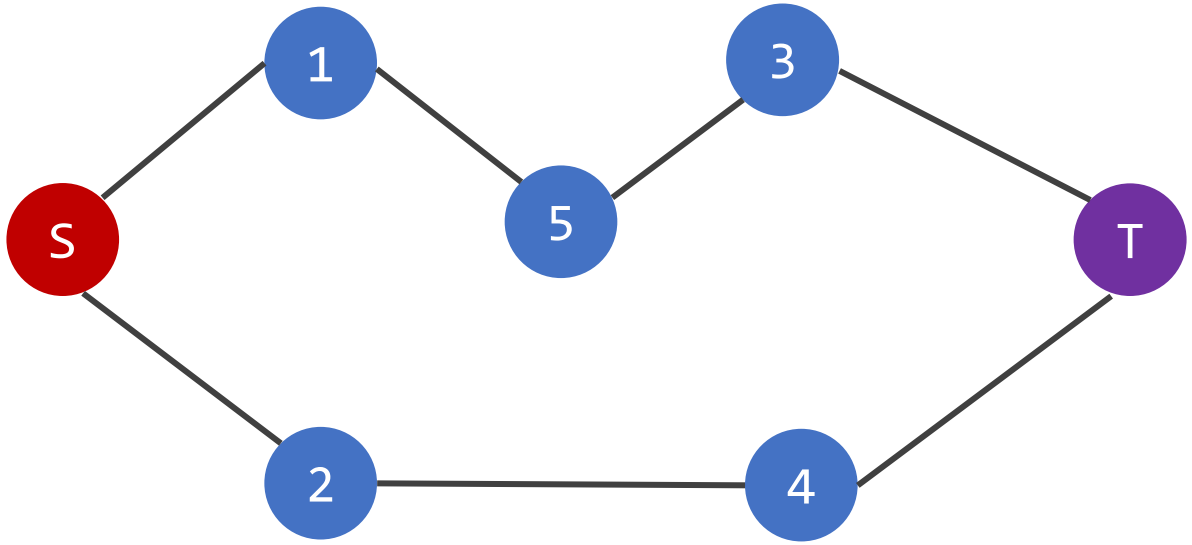
P0421. 抓住那头牛（双向队列优化）



可达信奥-小武老师-keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

双向BFS



求S-T的最短路，交替节点搜索（**一次正向节点，一次反向节点**）时

Step 1 : S –> 1 , 2

Step 2 : T –> 3 , 4

Step 3 : 1 –> 5

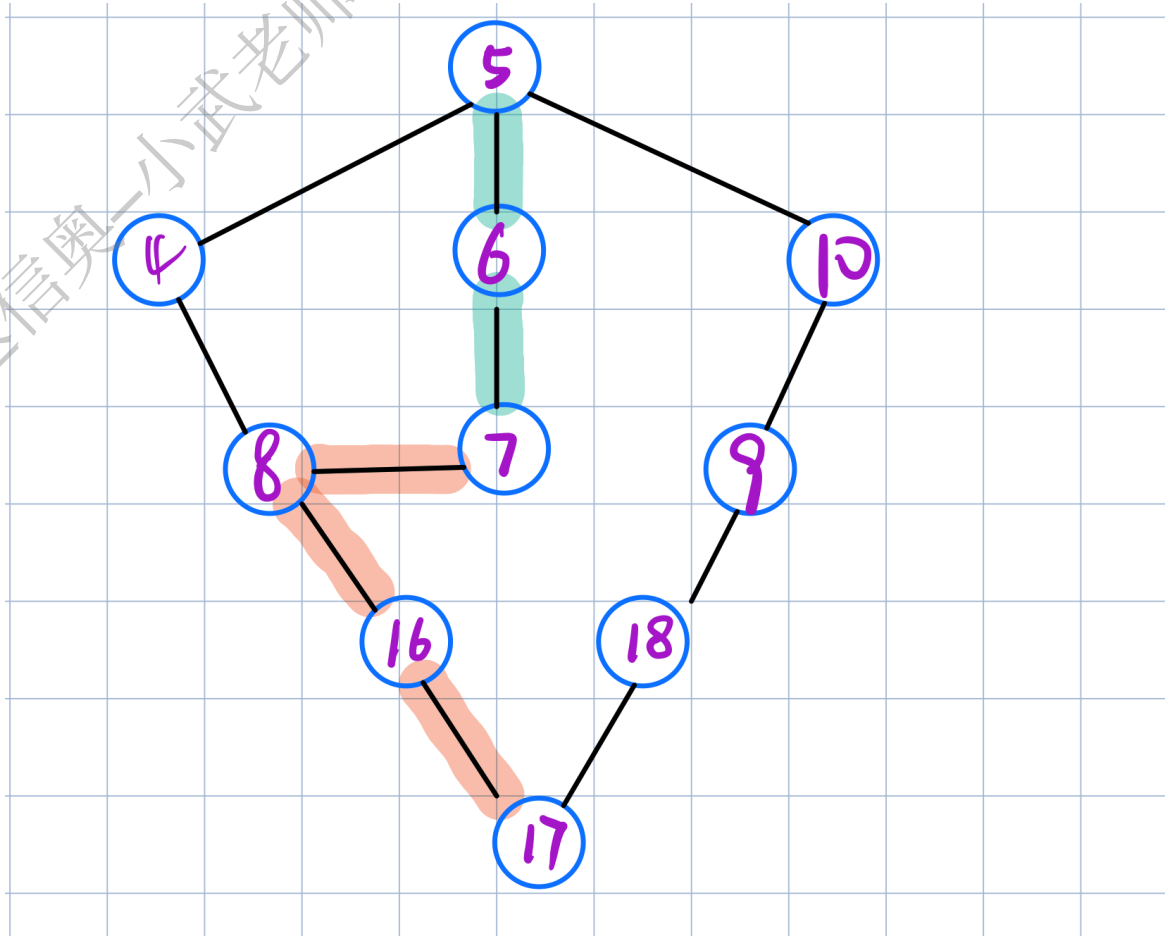
Step 4 : 3 –> 5　返回最短路为4，错误的，事实是3，S-2-4-T

可达信奥_小武老师_keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

双向BFS

P0420 走迷宫 （双向队列优化）



正确做法的是交替逐层搜索，保证了不会先遇到非最优解就跳出，而是**检查完该层所有节点，得到最优值**。

可达信奥_小武老师_keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

双向BFS

P0421. 抓住那头牛（双向队列优化）

尝试写一下

可达信奥_小武老师_keda.ac
可达信奥_小武老师_keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

双向BFS

P0436. 八数码难题

经典bfs题目，直接搜会很麻烦，中间会出现大量重复的状态，会浪费很多时间，因此为了去除重复状态，优化搜索效率，可以把这个三行转换成一个9位的整数。然后会发现这样转换后，每一种状态对应唯一的一个整数，因此队列中只需要保存整数（状态）即可，对于每一个整数都考虑是由哪一个整数转换得到的，同时相应步数加1，map会实现相同状态下的自动去重，十分方便，效率就会大大提升，但是在搜索的时候，还是需要转换回矩阵去判断0该往哪走。提供单向bfs和双向bfs的思路(单向bfs时间：31个测试点7962ms)(双向bfs时间：31个测试点298ms)快了20多倍

可达信奥_小武老师_keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

BFS总结

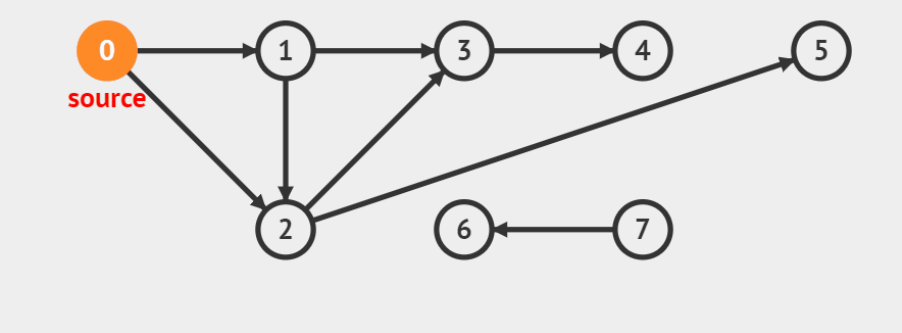
1、对于数据结构和算法的学习，难的不是掌握原理，而是能灵活地将各种场景和问题抽象成对应的数据结构和算法。比如迷宫可以抽象成图，走迷宫可以抽象成搜索算法。那么，如何将迷宫抽象成一个图？即如何在计算机中存储一个迷宫？

2、广搜原理简单，代码实现复杂。深搜适合递归实现，算法原理不易理解，但代码简洁。

3、广搜和深搜的算法时间复杂度都是O(V+E),空间复杂度都是O(v)

4、广搜需要用到队列这种数据结构，尽量手写队列。

5、广搜优化，双向搜索



可达信奥_小武老师-keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

课后习题与实验

Talk is cheap, show me the code !

可达信奥_小武老师_keda.ac
可达信奥_小武老师_keda.ac

声明：本课件及视频版权归小武老师所有，禁止任何组织及个人分发、抄袭、售卖等，违者将追究其法律责任！

可达信奥_小武老师_keda.ac

下节课见啦！

可达信奥_小武老师_keda.ac