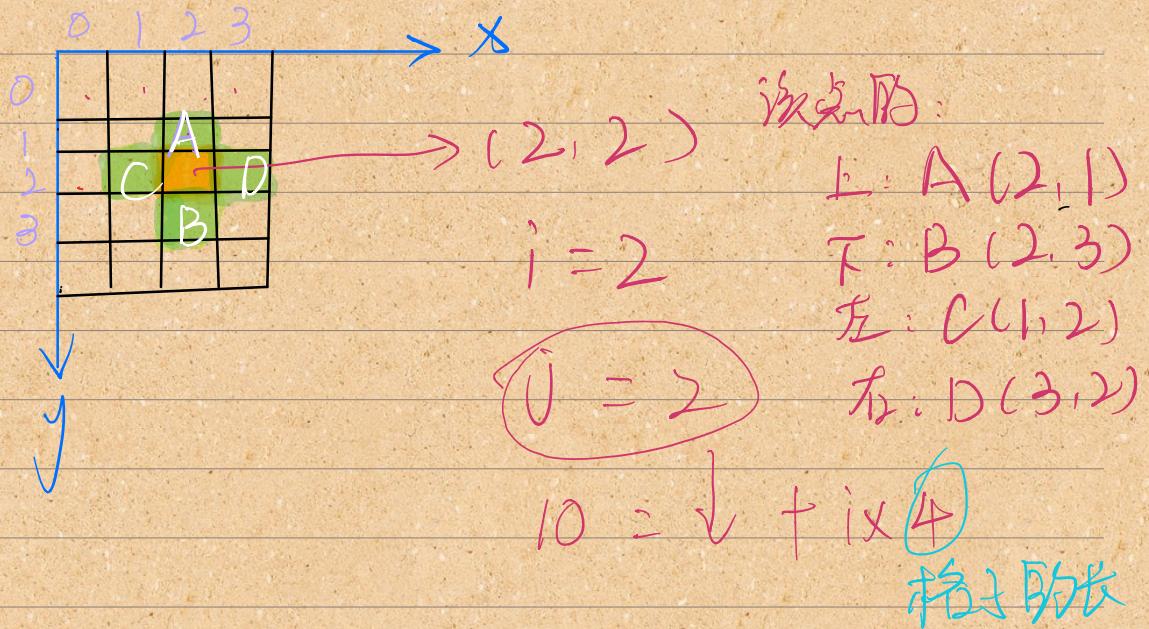


广度优先搜索 queue
 深度优先搜索 stack

?



1. 为什么用 queue (队列) 就是广度优先？

队列是一种先进先出的数据结构。在我们进行 while 循环时，通过 shift 取出队列中的第一个，就说明我们每次都是从头开始搜索下一层的，也就具有横向搜索（即广度优先）。

从左到右，从上到下，从内到外。

每次搜索完成一个点之后，就会接着去搜索下一级的点，直到把这一层次的点都搜索完毕。

如果是栈结构，先进后出，则为深度。

因为每次都是抓着最后一个点，直到搜至粗枝，才结束最后一个点。

启发式搜索：

能找到最优路径的叫 A*

A star

找不到的叫 A.

A 星算法是 A 算法的一种特例。

Tip:

splice 算法复杂度为 $O(n)$

⇒ 替代方案：
 $arr[min] = arr[max]$
 $arr.pop()$

复杂度：
 $O(1)$

两个点之间的距离？



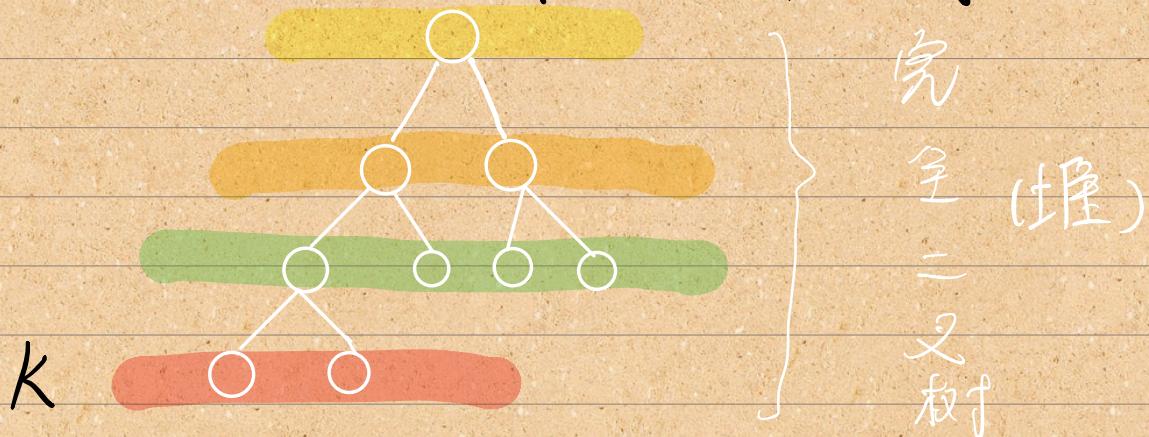
(x_1, y_1) \rightarrow

$$\text{distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

二叉堆 ?

二叉堆：是一个完全二叉树。

完全二叉树：若设二叉树的深度为 K ，除第 K 层外，其它各层 ($1 \sim K-1$) 层的结点数都达到最大，并且 K 层的所有结点都集中在左侧。



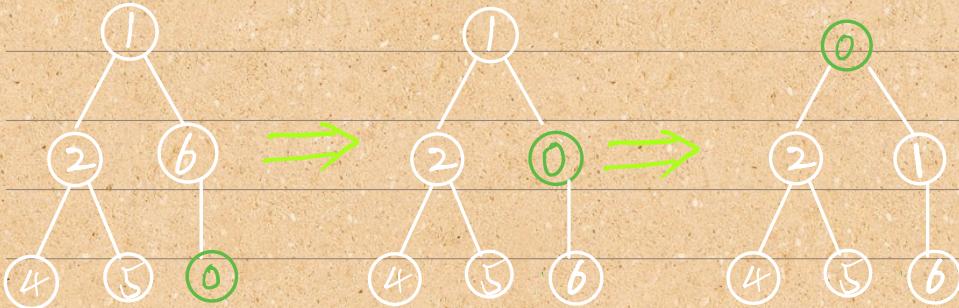
「最大堆」：根节点（堆顶）最大，子节点要小于其父节点。
「最小堆」：反之亦然。

TIP：这里的大小其实并不是传统意义上的大小，
描述为优先级会更准确。

二叉堆的操作：

1. 插入：

二叉堆的节点插入位置是从最后一个节点开始的。



插入至最后一个节点之后会循环与父节点比较大小，
直至符合其堆的特质结束

2. 删除：

把最后一个节点的值赋给需要删除的节点
位置，通过比较其左右子节点，不断下沉完成删除

3. 构建二叉堆