宽度优先搜索及其优化

~~大家都会*2~~

原理

宽度优先搜索（BFS）是常和深搜放在一起提及的算法。
同样是一种搜索方法，与深搜优先往深处前进不同，访问完当前节点后不会立刻去访问它的子节点，而是去访问“同一层”的另一个节点。在搜索树中表现出来就是“一层一层”地访问。
但“同层”的关系一般不会直接建立出来，这时候我们需要一种方法把一整层的节点先记下来。这种方法就是队列。
考虑维护一个队列，每次访问出队的节点，访问完之后将该节点的所有子节点入队。
当一层访问完之后，队列中剩下的是下一层的节点。
和深搜自然的栈结构不同，宽搜的队列需要自己手动维护。

例题

比较模板的就是无权图的最短路。

优化

和深搜一样，裸宽搜基本没什么用。

循环队列

应该算常规操作吧…
不过还是想列出来。毕竟挺有用的。

状态压缩

准确的说不是宽搜特有的优化策略。深搜也用得到。
因为搜索是对状态进行的操作，所以如何表示一个状态是很重要的一个问题。
对于许多题目而言，状态的存储可能会占据很大的空间，并且对于后期进行状态间的判断也带来相当大的麻烦。基于此，需要对状态进行压缩表示。
例如一个大数可以进行取模操作。

剪枝

不管是深搜还是宽搜，合理地剪枝总是一种简化的好方法。
当然也包含A*算法

双向宽搜

考虑到随着层数的增加，单层的节点数越来越多，所需的队列空间也越来越大。

为了解决这个问题，我们可以从起始状态和目标状态一起进行宽搜。
当状态相遇时搜索结束。