广度优先搜索BFS（Breadth First Search）也称为宽度优先搜索，它是一种先生成的结点先扩展的策略。

      在广度优先搜索算法中，解答树上结点的扩展是按它们在树中的层次进行的。首先生成第一层结点，同时检查目标结点是否在所生成的结点中，如果不在，则将所有的第一层结点逐一扩展，得到第二层结点，并检查第二层结点是否包含目标结点，……，对层次为n+1的任一结点进行扩展之前，必须先考虑层次完层次为n的结点的每种可能的状态。因此，对于同一层结点来说，求解问题的价值是相同的，可以按任意顺序来扩展它们。通常采用的原则是先生成的结点先扩展。

      为了便于进行搜索，要设置一个表存储所有的结点。由于在广度优先搜索算法中，要满足先生成的结点先扩展的原则，所以存储结点的表一般采用队列这种数据结构。

      在编写程序时，可用数组q模拟队列。front和rear分别表示队头指针和队尾指针，初始时front=rear=0。

      元素x入队操作为  q[rear++]=x;

      元素x出队操作为  x =q[front++];

      广度优先搜索算法的搜索步骤一般是：

      （1）从队列头取出一个结点，检查它按照扩展规则是否能够扩展，如果能则产生一个新结点。

       （2）检查新生成的结点，看它是否已在队列中存在，如果新结点已经在队列中出现过，就放弃这个结点，然后回到第（1）步。否则，如果新结点未曾在队列中出现过，则将它加入到队列尾。

      （3）检查新结点是否目标结点。如果新结点是目标结点，则搜索成功，程序结束；若新结点不是目标结点，则回到第（1）步，再从队列头取出结点进行扩展。

      最终可能产生两种结果：找到目标结点，或扩展完所有结点而没有找到目标结点。

      如果目标结点存在于解答树的有限层上，广度优先搜索算法一定能保证找到一条通向它的最佳路径，因此广度优先搜索算法特别适用于只需求出最优解的问题。当问题需要给出解的路径，则要保存每个结点的来源，也就是它是从哪一个节点扩展来的。

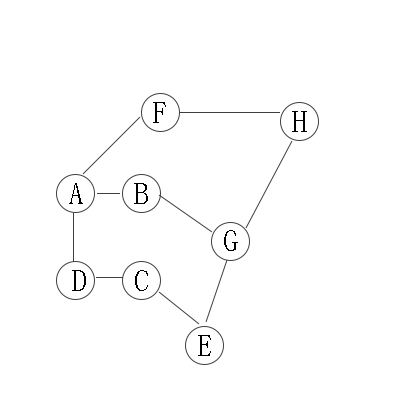
**深度优先遍历(Depth First Search)的主要思想是：**

　　　　1、首先以一个未被访问过的顶点作为起始顶点，沿当前顶点的边走到未访问过的顶点；

　　　　2、当没有未访问过的顶点时，则回到上一个顶点，继续试探别的顶点，直至所有的顶点都被访问过。

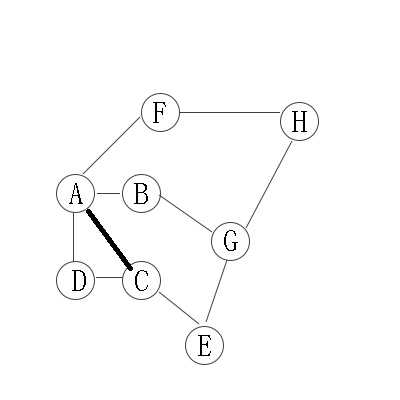
**1.1　无向图的深度优先遍历图解**

**以下"无向图"为例：**



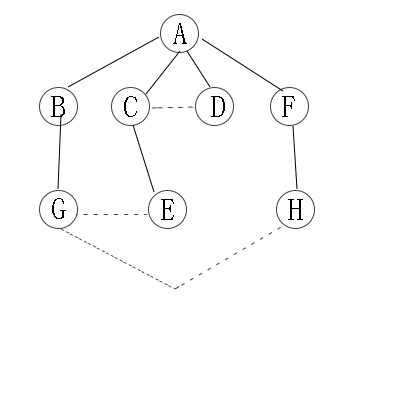
### 广度优先遍历(Depth First Search)的主要思想是：类似于树的层序遍历。

**2.1　无向图的广度优先遍历图解：**



从A开始，有4个邻接点，“B，C，D，F”，这是第二层；

在分别从B，C，D，F开始找他们的邻接点，为第三层。以此类推。



因此访问顺序是：**A -> B -> C -> D -> F -> G -> E -> H**

贪心算法（又称贪婪算法）是指，在对[问题求解](https://baike.baidu.com/item/%E9%97%AE%E9%A2%98%E6%B1%82%E8%A7%A3/6693186)时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，算法得到的是在某种意义上的局部最优解[1]  。

贪心算法不是对所有问题都能得到整体最优解，关键是贪心策略的选择。也就是说，不从整体最优上加以考虑，做出的只是在某种意义上的局部最优解[1]  。

使用条件

[编辑](javascript:;)

利用贪心法求解的问题应具备如下2个特征[4]  。

**1、贪心选择性质**

一个问题的整体最优解可通过一系列局部的最优解的选择达到，并且每次的选择可以依赖以前作出的选择，但不依赖于后面要作出的选择。这就是贪心选择性质。对于一个具体问题，要确定它是否具有贪心选择性质，必须证明每一步所作的贪心选择最终导致问题的整体最优解[4]  。

**2、最优子结构性质**

当一个问题的最优解包含其子问题的最优解时，称此问题具有最优子结构性质。问题的最优子结构性质是该问题可用贪心法求解的关键所在。在实际应用中，至于什么问题具有什么样的贪心选择性质是不确定的，需要具体问题具体分析[4]  。

解题策略

[编辑](javascript:;)

贪心算法不从整体最优上加以考虑，所做出的仅是在某种意义上的局部最优选择。使用贪心策略要注意局部最优与全局最优的关系，选择当前的局部最优并不一定能推导出问题的全局最优。贪心策略解题需要解决以下两个问题：[5] 

1、该问题是否适合使用贪心策略求解，也就是该问题是否具有贪心选择性质[5]  ；

2、制定贪心策略，以达到问题的最优解或较优解[5]  。

要确定一个问题是否适合用贪心算法求解，必须证明每一步所作的贪心选择最终导致问题的整体最优解。证明的大致过程为：首先考察问题的一个整体最优解，并证明可修改这个最优解，使其以贪心选择开始，做了贪心选择后，原问题简化为规模更小的类似子问题。然后用数学归纳法证明通过每一步做贪心选择，最终可得到问题的整体最优解[5]  。

存在问题

[编辑](javascript:;)

贪心算法也存在如下问题：[6] 

1、不能保证解是最佳的。因为贪心算法总是从局部出发，并没从整体考虑[6]  ；

2、贪心算法一般用来解决求最大或最小解[6]  ；

3、贪心算法只能确定某些问题的可行性范围[6]  。

应用实例

[编辑](javascript:;)

例如，平时购物找零钱时，为使找回的零钱的硬币数最少，不要求找零钱的所有方案，而是从最大面值的币种开始，按递减的顺序考虑各面额，先尽量用大面值的面额，当不足大面值时才去考虑下一个较小面值，这就是贪心算法[7]  。

