广度优先遍历

bfs总是先访问完同一层的结点，然后才继续访问下一层结点，它最有用的性质是可以遍历一次就生成中心结点到所遍历结点的最短路径

**import** queue  
**def** bfs(adj, start):  
 visited = set()  
 q = queue.Queue()  
 q.put(start)  
 **while not** q.empty():  
 u = q.get()  
 print(u)  
 **for** v **in** adj.get(u, []):  
 **if** v **not in** visited:  
 visited.add(v)  
 q.put(v)  
  
  
graph = {1: [4, 2], 2: [3, 4], 3: [4], 4: [5]}  
bfs(graph, 1)

1 创建一个队列，遍历的起始点放入队列

2 从队列中取出一个元素，打印它，并将其未访问过的子结点放到队列中

3 重复2，直至队列空

时间复杂度：基本与图的规模成线性关系了，比起图的其它算法动不动就O(n^2)的复杂度它算是相当良心了

空间复杂度：我们看到程序中使用了一个队列，这个队列会在保存一层的结点，当图规模很大时占用内存还是相当可观的了，所以一般会加上一些条件，比如遍历到第N层就停止