广度优先搜索算法

171491103 米日阿地·买买提明 硬件一班

1. **实验题目**

广度优先搜索算法（BFS）

1. **算法简介**

广度优先搜索算法（英语：Breadth-First-Search，缩写为BFS），又译作宽度优先搜索，或横向优先搜索，是一种图形搜索算法。简单的说，BFS是从根节点开始，沿着树的宽度遍历树的节点。如果所有节点均被访问，则算法中止。

三．**算法实现步骤**

（1）首先将根节点放入队列中。

（2）从队列中取出第一个节点，并检验它是否为目标。

（3）如果找到目标，则结束搜索并回传结果。

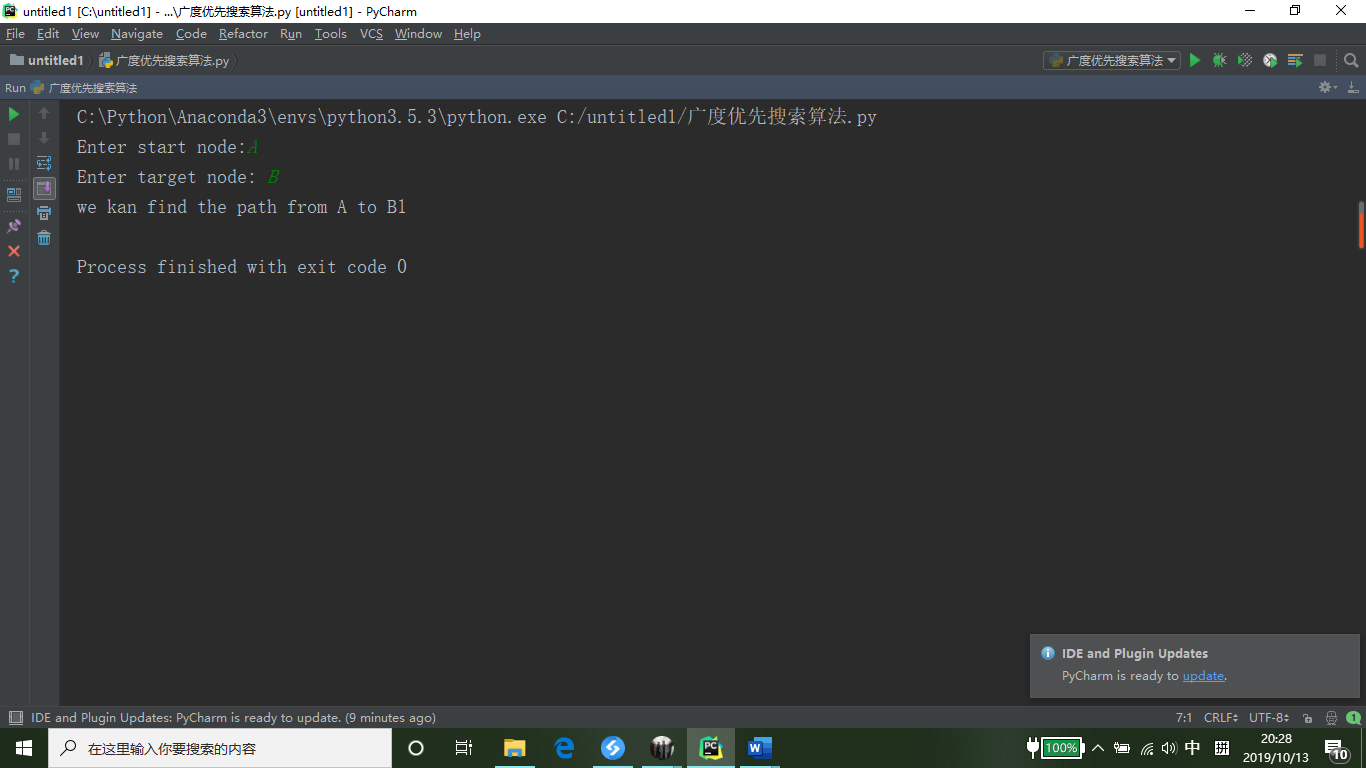
（4）否则将它所有尚未检验过的直接子节点加入队列中。

（5）若队列为空，表示整张图都检查过了——亦即图中没有欲搜索的目标。结束搜索并回传“找不到目标”。

**三：实验代码**

1. from collections import deque  
     
   start\_node = input("Enter start node:")#获得查询起点  
   target\_node = input("Enter target node: ")#获得查询终点  
     
   #生成图的字典  
   graph = {}  
   graph['A'] = ['B1', 'B2','B3']  
   graph['B1'] = ['C1','C2','D1']  
   graph['B2'] = ['C1', 'C3', 'C4']  
   graph['B3'] = ['C5']  
   graph['C1'] = ['C3', 'D1']  
   graph['C2'] = ['D1']  
   graph['C3'] = ['D1']  
   graph['C4'] = ['C3', 'C5']  
   graph['C5'] = []  
   graph['D1'] = []  
     
   #定义查找函数  
   def search(name): #name是查找起点  
    search\_queue = deque() #船舰查询队列  
    search\_queue += graph[name]  
    searched = []  
    while search\_queue:  
    node = search\_queue.popleft()  
    if node not in searched:  
    print("we kan find the path from {} to {}".format('A',node))  
    return True  
    else:  
    search\_queue += graph[node]  
    searched.append(node)  
    print("No path from {0} to {1}",format('A',node))  
    return False  
     
     
     
   def node\_is\_target(node):  
    return node == target\_node  
     
   search(start\_node)

实验结果

**五：实验总结**

如果有两个节点同时指向下一个节点，如C1、C4都指向C3，这时C3再队列中会显示两次，但对于判断来说只需判断一次

如果图中存在无向图（无向图即双向关系），这种情况下会生成循环队列，使用此标记可以避免循环队列带来的冗余计算