A screenshot of a computer program

Description automatically generated

graphSearch.py

import copy  
import heapq  
import numpy as np  
from Project3.lwnode import \*  
  
class graphSearch:  
 def \_\_init\_\_(self, node\_type):  
 self.node\_type = node\_type  
 # The heap for nodes to visit is kept here since it's a common component across different  
 # search strategies  
 self.heap = []  
  
 def run(self, seed, endnode=None):  
 heap = []  
 start\_node = self.node\_type(nd=seed)  
 heapq.heappush(heap, start\_node)  
 while heap:  
 node = heapq.heappop(heap)  
 if self.marked[node.nd]:  
 continue  
 self.mark(node)  
 self.setPointer(node, node.pr)  
 if node.nd == endnode:  
 return self.trace(node.nd, seed), node.cost  
 for neighbor in self.findNeibs(node):  
 if not self.marked[neighbor.nd]:  
 heapq.heappush(heap, neighbor)  
 return None  
  
 def trace(self, end\_nd, seed):  
 path = []  
 current = end\_nd  
 while current != seed:  
 path.append(current)  
 current = self.getPointer(current)  
 if current is None: # Safety check in case of disconnected nodes  
 return []  
 path.append(seed)  
 return path # Reverse the path to start from seed to end\_nd