**110550126 曾家祐**

**Homework 2: Route Finding**

**Part I. Implementation :**

* **Part 1: Breadth-first Search：bfs.py -> bfs()**

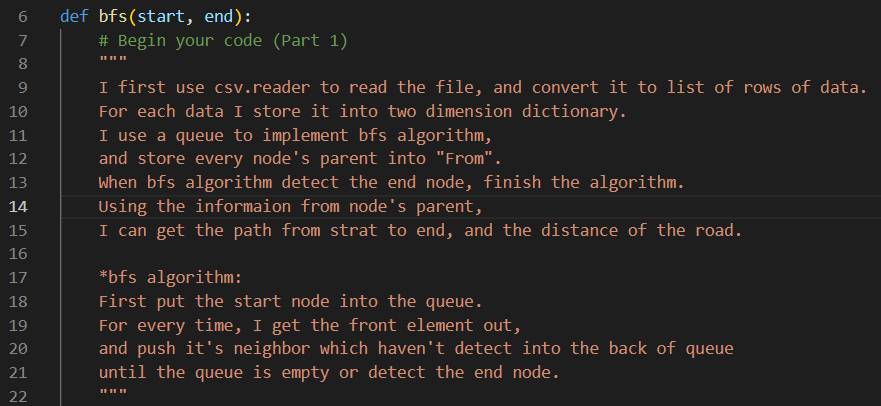
**1. Read data: I use csv.reader to read file, and make each data (start, end, distance, speed limit)into dictionary (edges[start][end] = (distance, speed limit))**

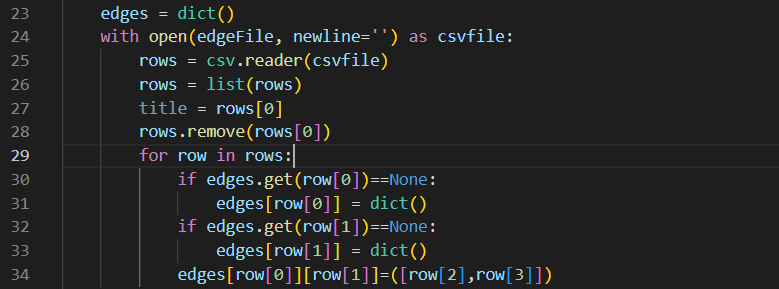
**2.** **bfs algorithm: First put the start node into the queue.**

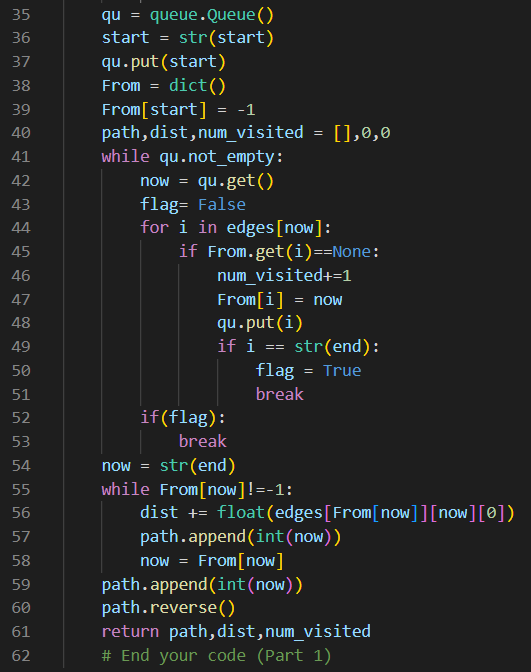
**For every time, I get the front element out, and push its neighbor which haven't detect into the back of queue.**

**until the queue is empty or detect the end node.**

**3. get result in the bfs: I store every node’s parent, from the information, I can get the path, and calculate the distance**

****

****

****

* **Part 2: Depth-first Search: dfs\_stack.py -> dfs()**

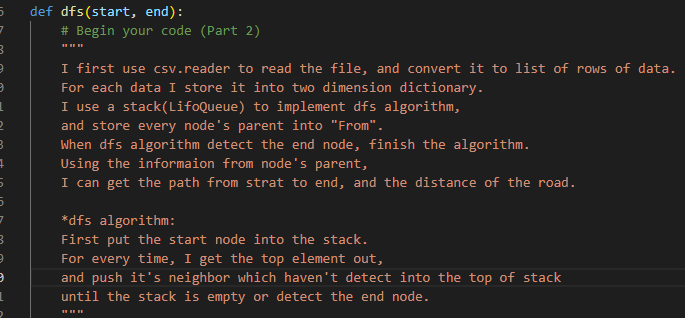
**1. Read data: I use csv.reader to read file, and make each data (start, end, distance, speed limit)into dictionary (edges[start][end] = (distance, speed limit))**

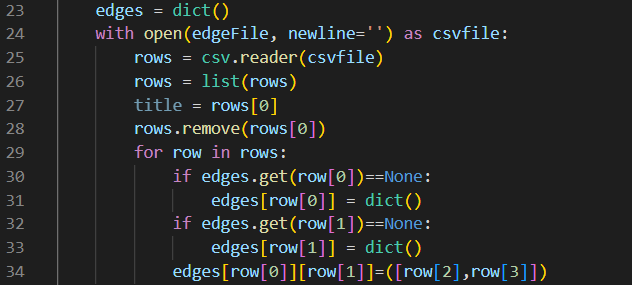
**2.** **dfs algorithm: First put the start node into the stack.**

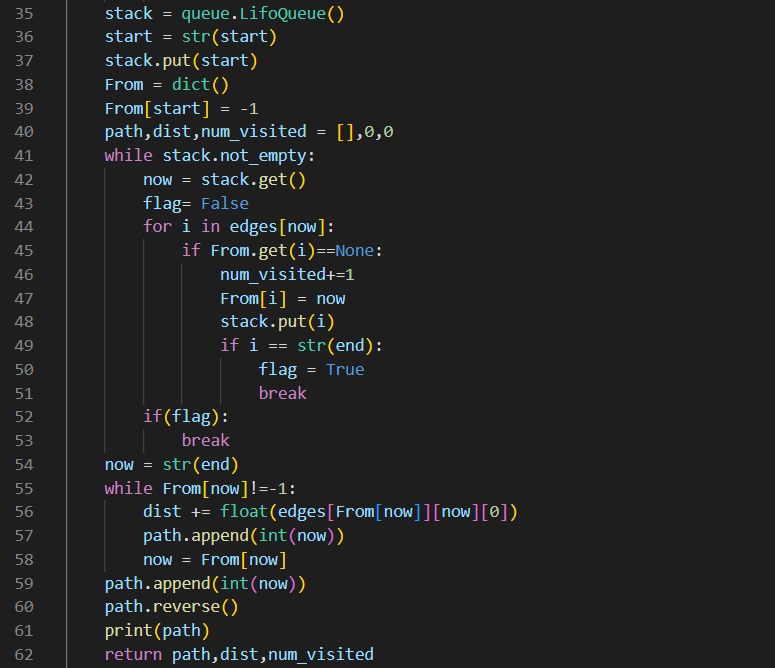
**For every time, I get the top element out, and push its neighbor which haven't detect into the top of stack.**

**until the stack is empty or detect the end node.**

**3. get result in the dfs: I store every node’s parent, from the information, I can get the path, and calculate the distance.**







* **Part 3: Uniform Cost Search**

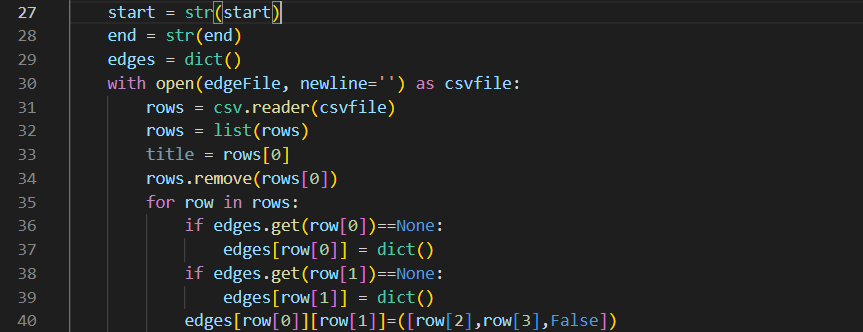
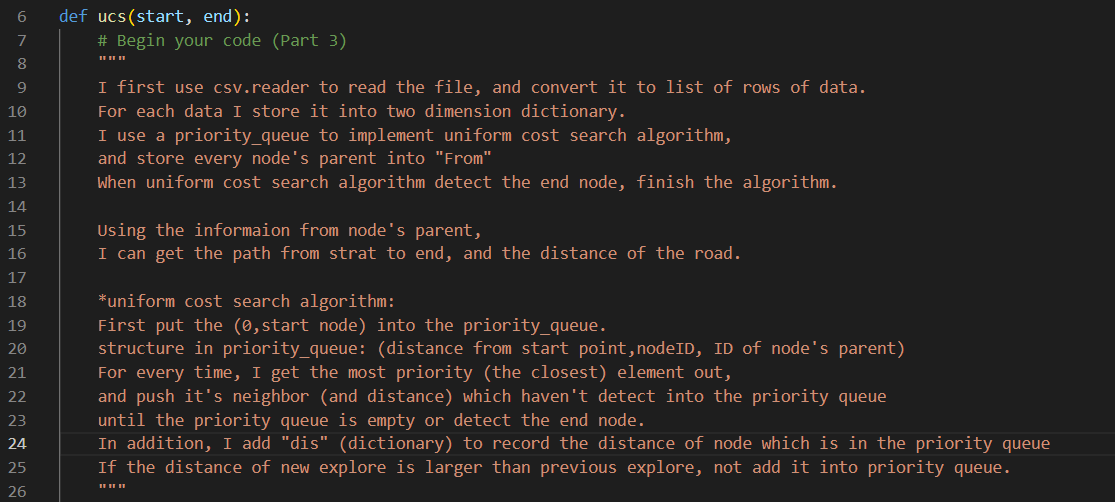
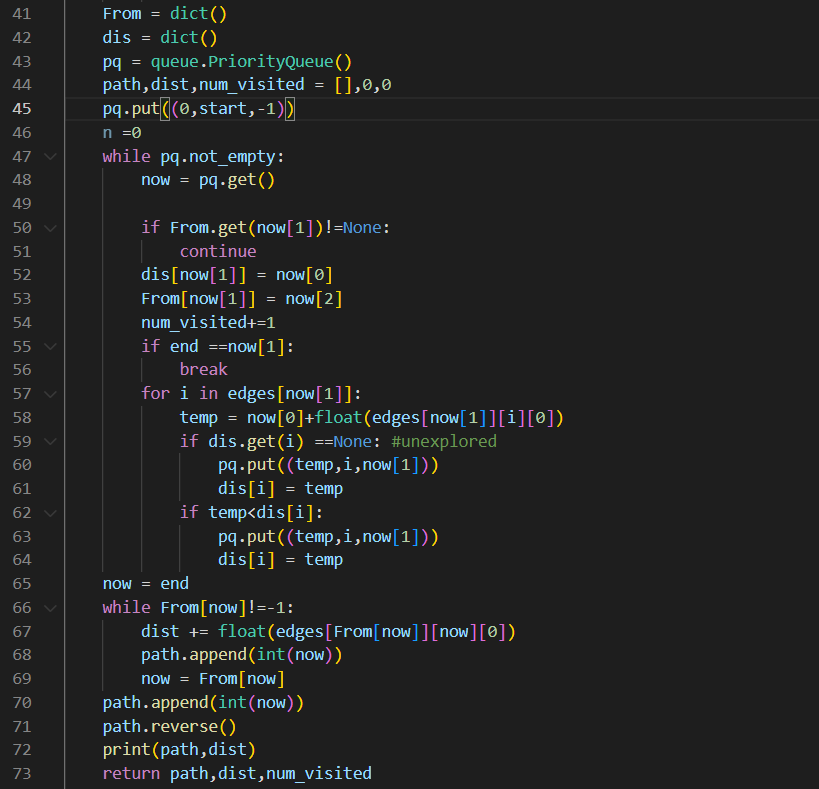
**1. Read data: I use csv.reader to read file, and make each data (start, end, distance, speed limit)into dictionary (edges[start][end] = (distance, speed limit)),**

**2.** **dfs algorithm: First put (0,start node, -1 )(distance, nodeID, parentID)into the priority\_queue.**

**For every time, I get the closest element out, and push its neighbor which haven't detect into priority\_queue.**

**until the priority\_queue is empty or detect the end node.**

**3. get result in the ucs: I store every node’s parent, from the information, I can get the path, and calculate the distance.**

****

* **Part 4: A\* Search**

**1. Read data: I use csv.reader to read file, and make each data (start, end, distance, speed limit)into dictionary (edges[start][end] = (distance, speed limit)) and (startID, endID, distance) into (heur[startID][endID] = distance)**

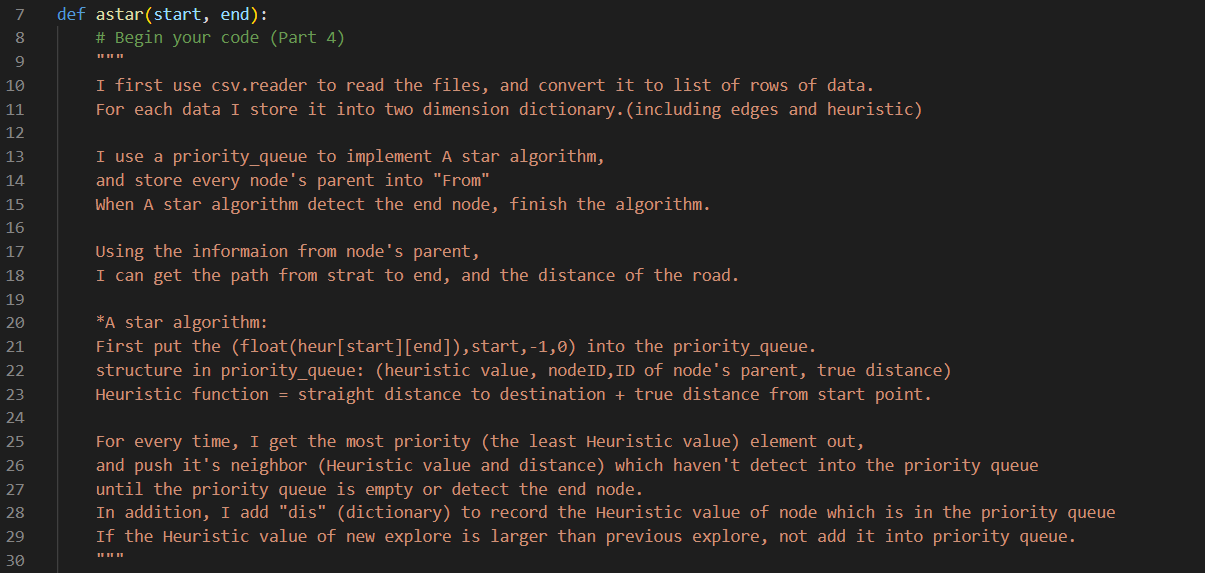
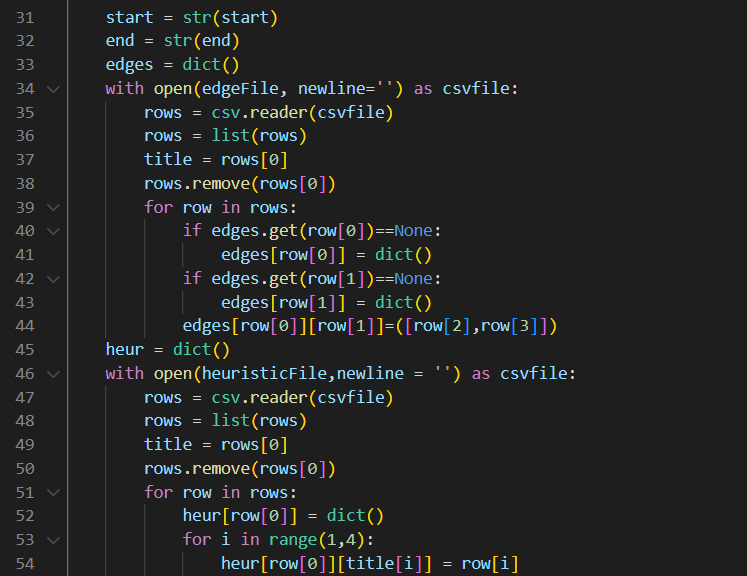
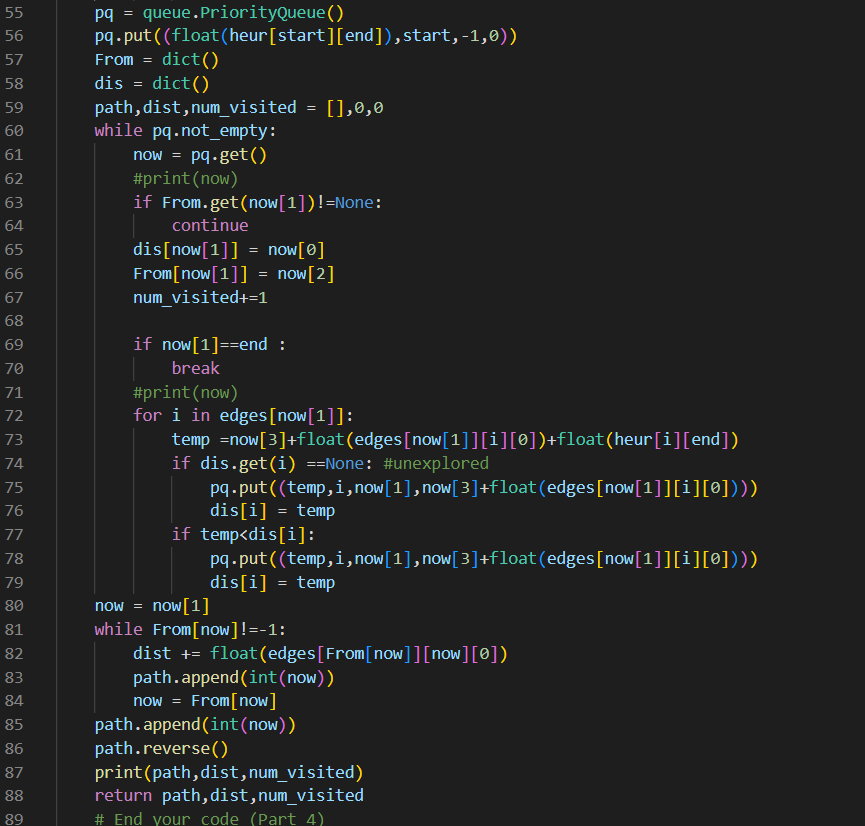
**2.** **A star algorithm: First put (heur[startID][endID],start node,-1,0)(heuristic value, nodeID, parentID, true distance)into the priority\_queue.**

**For every time, I get the closest element out, and push its neighbor which haven't detect into priority\_queue.**

**until the priority\_queue is empty or detect the end node.**

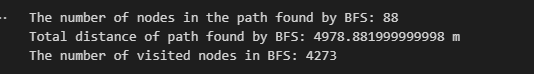
**\* heuristic function: straight distance to end point + true distance from start point.**

**3. get result in the A star: I store every node’s parent, from the information, I can get the path, and calculate the distance.**

* ****  

**Part II. Results & Analysis**

* + **Test1: from National Yang Ming Chiao Tung University (ID: 2270143902) to Big City Shopping Mall (ID: 1079387396)**

****

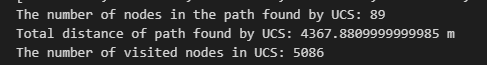
****

**DFS(stack):**

****

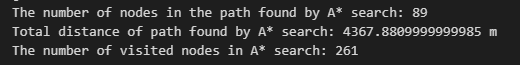
****

**UCS:**

****



**A star:**

****



**A star time:**

****

* + **Test2: from Hsinchu Zoo (ID: 426882161) to COSTCO Hsinchu Store (ID:** **1737223506)**

**BFS:**

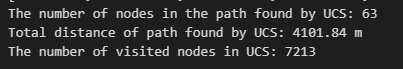
****

****

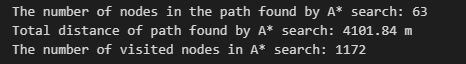
**DFS(stack):  
**

****

**UCS:**

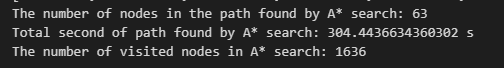
****

**A star:**

****

****

**A star time:**

****

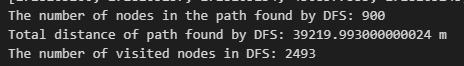
**Test3: from National Experimental High School At Hsinchu Science Park (ID: 1718165260) to Nanliao Fighing Port (ID: 8513026827)**

**BFS:**

****

****

**DFS(stack):**

****



**UCS:**

****

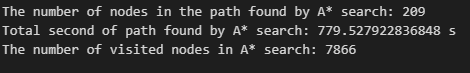


**A star:**

****



**A star time:**

****



**Part III. Question Answering**

1. **Please describe a problem you encountered and how you solved it.**

The problem I encounter is ucs and A star algorithm, these two algorithm is new for me. So I did a lot of search on the internet, and try to understand how it works. After know how why the heuristic function work and come up with the heuristic function that works.

1. **Besides speed limit and distance, could you please come up with another attribute that is essential for route finding in the real world? Please explain the rationale.**

When the time is commute time, the traffic is heavy on some road. Although the distance is shorter, but the time I travel is longer, so I think the time and the traffic is busy or not is also important to consider. Furthermore, traffic light also need to be considered because some small road have shortest distance, but need to wait longer red light.

1. **As mentioned in the introduction, a navigation system involves mapping, localization, and route finding. Please suggest possible solutions for mapping and localization components?**

Mapping: To let computer know the map, we can use the help from satellite which can overview the road and the data from google map etc. to construct a graph.

Localization: To locate where the start(user) and destination are. This need the help of GPS to know where the start point is. With the information, we can locate the start and end point and use in path finding system.

1. **The estimated time of arrival (ETA) is one of the features of Uber Eats. To provide accurate estimates for users, Uber Eats needs to dynamically update ETA based on their mechanism. Please define a dynamic heuristic equation for ETA and explain the rationale of your design. Hint: You can consider meal prep time, delivery priority, multiple orders, etc.**

The element we need to consider is everything the deliver may encounter. Ex: reach the restaurant before meal prepared and deliver need to wait for the meal. Whether the deliver have multiple order and the priority in these order, we assume each delivery need 15 mins0

ETA = max(to restaurant time, meal prep time) + priority rank \*15

**Bonus: Part 6: Search with a different heuristic**

1. **Read data: I use csv.reader to read file, and make each data (start, end, distance, speed limit)into dictionary (edges[start][end] = (distance, speed limit)) and (startID, endID, distance) into (heur[startID][endID] = distance)**

**\*convert speed from km/h into m/sec**

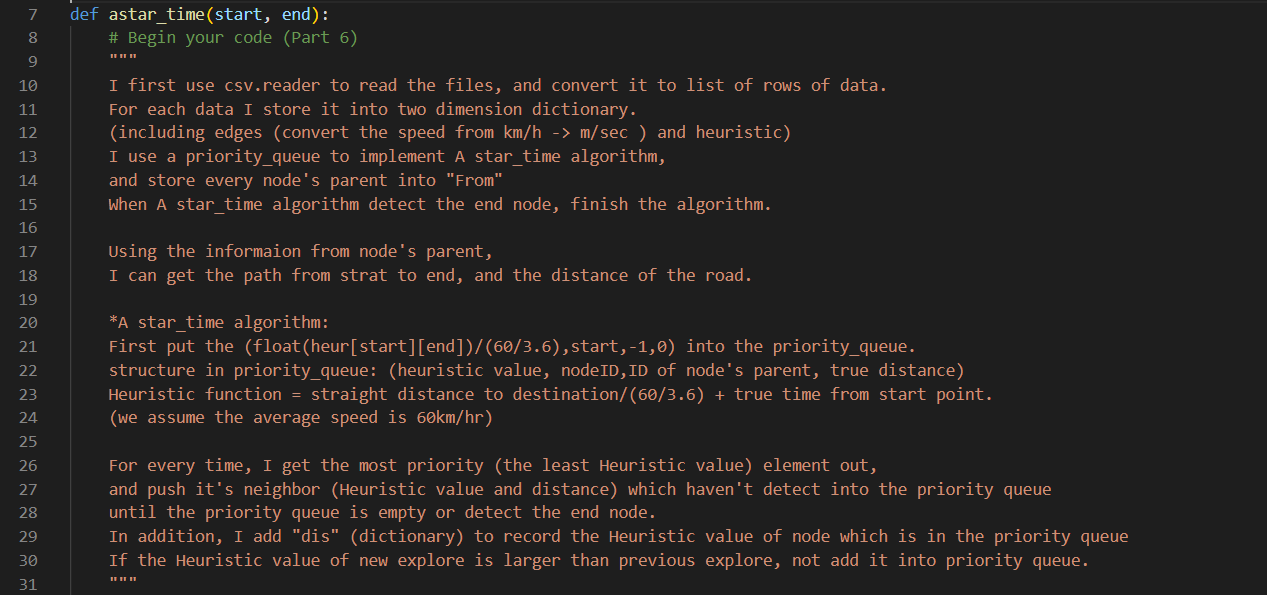
**2.** **A star time algorithm: First put (heur[startID][endID]/(60/3.6),start node,-1,0)(heuristic value, nodeID, parentID, true time)into the priority\_queue.**

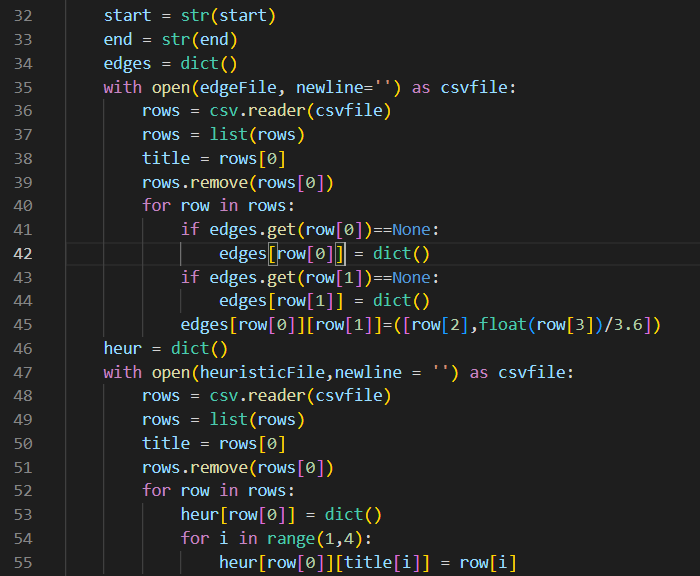
**For every time, I get the closest element out, and push its neighbor which haven't detect into priority\_queue.**

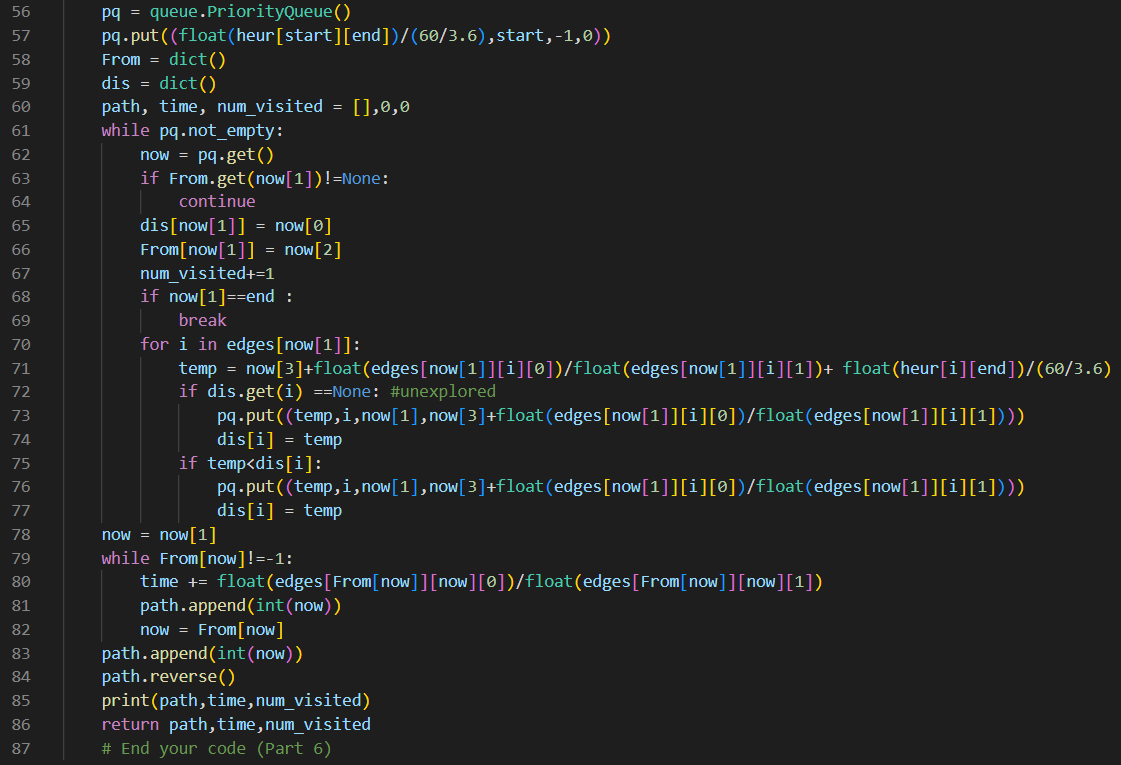
**until the priority\_queue is empty or detect the end node.**

**\* heuristic function: straight distance to end point/(60/3.6) + true time from start point. We assume the average is 60km/h**

**3. get result in the A star time : I store every node’s parent, from the information, I can get the path, and calculate the time.**

****

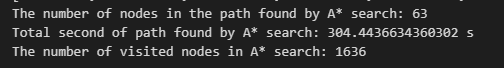
****

****

**Test1:**

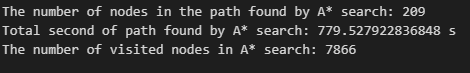
****

**Test2:**

****

****

**Test 3:**

****

