Final – Programming Assignment

Name: Vikhyat Dhamija
RUID: 194003013
NetID:vd283

Q1. Write a program that computes the "diameter" of a directed graph which is defined as the maximum-length shortest path connecting any two vertices. Estimate the runtime of your algorithm. Feel free to use the following datasets:

https://algs4.cs.princeton.edu/44sp/tinyEWD.txt

https://algs4.cs.princeton.edu/44sp/mediumEWD.txt

https://algs4.cs.princeton.edu/44sp/1000EWD.txt

https://algs4.cs.princeton.edu/44sp/10000EWD.txt

Solution:

Below is the time estimate , resulted when algorithm was run for the Datasets Provided:

S.No.	Dataset	Edges	Vertices	Time Taken(ms)	Order
1.	tinyEWD	15	8	0.0	E Log V * V
2.	mediumEWD	2546	250	939.1319751739502	E Log V * V
3.	1000EWD	16866	1000	23599.119901657104	E Log V * V
4.	10000EWD	123462	10000	2914599.505186081	E Log V * V

Algorithm Explanation:

1. Dijkstra algorithm was used for calculating the shortest distance to every other vertex in the graph from a single source .
2. So Dijkstra was made to work for every vertex of the graph
3. We have taken an initial variable Max which will compare itself with every shortest path , in order to at last give us the Maximum-Shortest Path which is the diameter of the graph as asked in the question.

Time estimation(theoretical):

In Dijkstra

1. First , we choose source
2. Relax all its connecting edges
3. Then perform insertion or decrease key on each edge depending on whether the edge for that vertex in the priority key exists or not (E*Log V as log V for all edges in total)
4. Then delete min to find shortest path to one vertex and then move on to other vertices connected/adjacent to that vertex (V * Log V as Log V for V vertices in total from one source Dijkstra)
5. This we repeat for all vertices from the source one by one
6. This all we repeat for all the vertices for running Dijkstra over all vertices.(make whole algorithm order multiplied by V)

Steps one to five belongs to regular Dijkstra where decrease key/insert key always happening for all edges so E Log V (Log V because binary heap is being used).

Now as the above is the complexity of the one source Dijkstra but for getting the diameter we have to calculate Shortest Paths from every vertex. So V* E Log V is the order. So Algorithm becomes of O(V * E Log V)

Results:

a. **Dataset 1**

```
C:\Users\vikhyat\data_structures_algo\Final_exam_dsa>python q1.py
The diameter is a path between the points-------with distance value :
3 ----> 1 ----- 1.86
The time taken for the algorithm to execute is :  0.0  miliseconds
```

b. **Dataset 2**

```
C:\Users\vikhyat\data_structures_algo\Final_exam_dsa>python q1.py
The diameter is a path between the points-------with distance value :
123 ----> 237 ----- 1.37466
The time taken for the algorithm to execute is :  939.1319751739502  miliseconds
```

c. **Dataset 3**

```
C:\Users\vikhyat\data_structures_algo\Final_exam_dsa>python q1.py
The diameter is a path between the points-------with distance value :
13 ----> 466 ----- 1.39863
The time taken for the algorithm to execute is :  23599.119901657104  miliseconds
```

d. **Dataset 4**

```
C:\Users\vikhyat\data_structures_algo\Final_exam_dsa>python q1.py
The diameter is a path between the points-------with distance value :
1471 ----> 5933 ----- 1.4439799999999998
The time taken for the algorithm to execute is :  2914599.505186081  miliseconds
```

The Order function is having variables E and V:

I have just estimated T4/T3 using theoretical EV Log V= 97 and according to our estimation of Time=123. So theoretical estimations giving true order of growth of almost same as there are some other factors also.

Q2. Write a program to find values of "A" and "M", with M as small as possible, such that the hash function (A*K) modulo M, for transforming the Kth letter of the alphabet into a table index produces distinct values for the keys S E A R C H X M P L.

Solution:

Hash Function is : (A*K) modulo M or (A*K)%M

As already explained in the program comments again explaining my Logic for this program:

When we use Modulo M i.e. the remainder function then obviously, we can expect 0 to M-1 Values as result .

Here in the question it is asked to put the 10 values in the memory table so that hash function should compute the unique indexes.

So K is fixed which is the position value of the given Keys in their alphabet order.

Now we have to check whether the computation of the hash values for all K given is unique or not. For this we have

Loop 1 for M Variation and we move from 10 as we need at least 10 bins

Then loop 2 inside for A variation for particular M Note we have varied A from 0 to M-1 only as A can be n M+(0 to M-1) = (0 to M-1) so if we now the values in the range (0 to M-1) we can know other values of A by adding any n times M.

As we had to select the minimum M so we ran Loop 1 and stop it when the unique values for that M detected for all Keys and appended all values of A in the list.

For Calculation of Unique Values:

1. In built method:
   Using set function

2. We can do O(N^2) kind of operation like in selection sort to take 1 value compare with all other then next and then with all others like this

3. We can use the N Log N Merge Sort and then look at all values in N order so total N Log N operation.

But as complexity improvement was not the objective of the problem I have sorted using Bubble sort and then scanned so mine O(n^2) but can be made N Log N using the Merge sort.

```
C:\Users\vikhyat\data_structures_algo\Final_exam_dsa>python q2.py
The results are :
For the smallest value of M which is :  20
The  values of A are :  [1, 3, 7, 9, 11, 13, 17, 19]
```