Team members: Allen Clement Justine Clement (G48263308), Yeshawini Balakrishnan (G45297250), Yashwanth Raj Varadharajan - (G47635180), Sai Nikhil Varada (G43353336)

Date : 11-01-2023

Project 3

1 Problem Statement

“Teleportation in Astro Haunted Galaxies”

You have a teleporter that can take you from galaxy i to galaxy j. The cost to teleport is given by c(i,j), which can be arbitrary. Some galaxies are “astro-haunted” – this is specified by a(i) which can be 0 or 1 (1 means that that galaxy is “astro-haunted”). Give a polynomial time algorithm that minimizes the cost of going from galaxy 1 to galaxy n, such that you pass through no more than k astro-haunted galaxies. (You can assume that galaxies 1 and n are not astro-haunted.)

2 Theoretical Analysis

The All Pairs Shortest Path method can be used to easily solve the base case. In the event that a galaxy is astro-haunted, APSP will not be applicable; however, in that scenario, c(i,j) provides a straightforward solution. D(i,j,0) = min {c(i,j); APSP of removing all AH galaxies}. There’s also a recursive algorithm in it where each calculation of D(i,j,m) takes O(n) time. So, the Time Complexity of the base case is O(n^3), and by combining it with the recursive portion, the total time complexity comes as O(k * n^3)

3 Experimental Analysis

3.1 Pseudocode:

```
Function min_cost(n, k, a, c):
   Create a 2D array dp of size (n + 1) x (k + 1)
   for j from 0 to k:
      dp[1][j] = 0
   for i from 2 to n:
      for j from 1 to k:
         if a[i] is 1:
            for i_prime from 1 to n:
               if a[i_prime] is 1:
                  dp[i][j] = Minimum of (dp[i][j], dp[i_prime][j - 1] + c[i_prime][i])
   Set min_cost1 to dp[n][k]
   Return min_cost1
Create an empty list a
Append 0 to list a
Read n from input  # Number of galaxies
Read k from input  # Maximum number of astro-haunted galaxies allowed
Create a 2D array c of size (n + 1) x (n + 1)
for i from 0 to n:
   Append a random value (0 or 1) to list a
Set the last element of list a to 0
for i from 1 to n:
   for j from (i + 1) to n:
      Set c[i][j] to a random integer between 1 and 10
Start a timer
Call min_cost(n, k, a, c) and store the result in min_cost
Print "Minimum cost to reach the galaxy", n, "with at most", k, "astro-haunted galaxies is:", min_cost
```

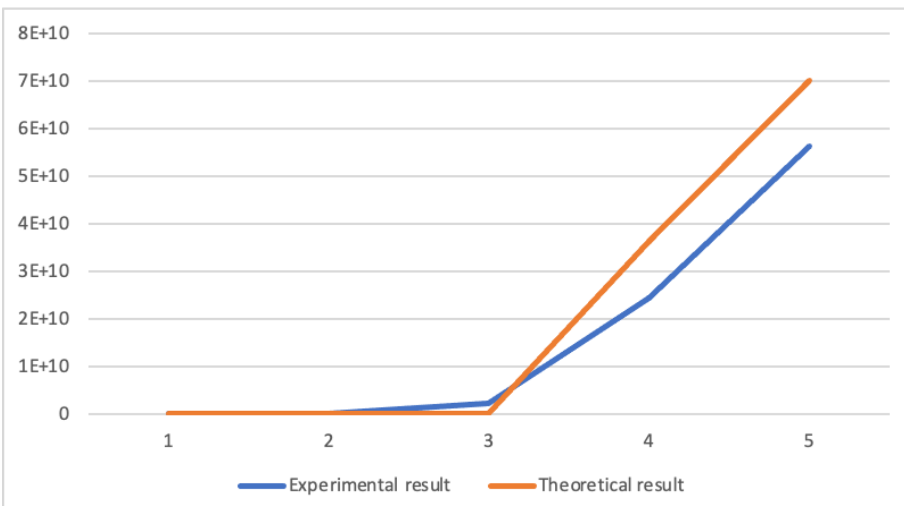
3.2 Data Normalization Notes

The theoretical value is close to the experimental one's and thus we do not need a scaling constant (the scaling constant found was insignificant - 1.24492)

3.3 Output Numerical Data

S.No	k	n	Experimental result	Theoretical result
1	4	10	59080000	4000
2	25	60	90030000	5400000
3	47	100	2274400000	47000000
4	290	500	24460690000	36250000000
5	70	1000	56230206000	70000000000
			17077761200	21260480800

3.4 Graph



3.5 Graph Observations

By observing the graph we can say that the lines of Theoretical Value and Experimental Value grow similarly. Hence we can say that we can find similar growth of values from the algorithm given. It is also observed that we don't need any scaling constant as the graph progresses similarly.

4 Conclusions

For the given algorithm I found the time complexity of O(k * n^3) and for that, I derived our theoretical value by applying this algorithm in a program I got our practical value. For the values I derived, a graph is drawn to show the comparison between them.

5 Github

https://github.com/2MrRobot2/GWU.git