Computational methods

Task 1

A screenshot of a graph

Description automatically generated

The cheapest route is the Delta, Alpha, Epsilon, Beta and gamma for a value of 69000 alien currency.

The reason why brute forcing is not optimal in this scenario is because you would need to go through the whole values available to you and check between all of them for the total of the whole trip. Yes this is the reason of brute forcing but that doesn’t help the cause of the scenery, it would be optimal use a different method.

Task 2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bubble Sorting | | | |  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 15 | 12 | 12 | 25 | 16 | 20 | 14 | 28 | 17 |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 12 | 15 | 12 | 25 | 16 | 20 | 14 | 28 | 17 |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 12 | 12 | 15 | 25 | 16 | 20 | 14 | 28 | 17 |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 12 | 12 | 15 | 16 | 25 | 20 | 14 | 28 | 17 |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 12 | 12 | 15 | 16 | 20 | 25 | 14 | 28 | 17 |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 12 | 12 | 15 | 16 | 20 | 14 | 25 | 28 | 17 |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 12 | 12 | 15 | 16 | 20 | 14 | 25 | 17 | 28 |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 12 | 12 | 15 | 16 | 14 | 20 | 17 | 25 | 28 |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 12 | 12 | 15 | 16 | 14 | 17 | 20 | 25 | 28 |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 12 | 12 | 15 | 14 | 16 | 17 | 20 | 25 | 28 |
|  |  |  |  |  |  |  |  |  |  |
| 10 | 12 | 12 | 14 | 15 | 16 | 17 | 20 | 25 | 28 |

C++

1. **int** main()
2. {
3. **int** arr[10] = { 10, 15, 12, 12, 25, 16, 20, 14, 28, 17 };

6. **for** (**int** i = 0; i < 10-1; i++)
7. {
8. **int** minIndex = i;
9. **for** (**int** j = i + 1; j < 10; j++)
10. {
11. **if** (arr[j] < arr[minIndex])
12. {
13. minIndex = j;
14. }
15. }
17. **int** temp = arr[minIndex];
18. arr[minIndex] = arr[i];
19. arr[i] = temp;
20. }
21. }

Worse case

Quadratic Time Complexity is related to the input list, if the list is sorted in the exact opposite order (O(n2)) would make it the longest way to sort it with the bubble sorting

StudySmarter Bubble Sort (Poor Efficiency on large datasets(26 November 2024))

Best care

The best case is a fully sorted list from the beginning without needing to do anything.

StudySmarter Bubble Sort (limitations of Bubble Sort (26 November 2024))

Task 3

A greedystrategy is a simple, intuitive algorithm that is used in optimization problems. The algorithm makes the optimal choice at each step as it attempts to find the overall optimal way to solve the entire problem.

Prseudocode

1. function Greedy(Graphic,source):
2. **for** each vertex v in Graphic.Vertices:
3. dist[v] ← INFINITY
4. prev[v] ← UNDEFINED
5. add v to Q
6. dist[source] ← 0
8. **while** Q is not empty:
9. u ← vertex in Q with min dist[u]
10. **remove** u from Q
12. **for** each neighbor v of u still in Q:
13. alt ← dist[u] + Graphic.Edges(u, v)
14. **if** alt < dist[v]:
15. dist[v] ← alt
16. prev[v] ← u
18. **return** dist[],prev[]

setting v as a number for the amount of vectors sets the algorithm to O(V2) the main reason for this happening is because of the vertex with the lowest distance need to look for the next vertex to go to and that makes O(V) time. Regarding it we need to do It for every vertex connected to the current one so we get time complexity O(V2) for the greedy algorithm Dijkstra’s strategy.

Time complexity do dijkstra’s algorithm W3Schools ( <https://www.w3schools.com/dsa/dsa_algo_graphs_dijkstra.php>) (26 November 2024)

Task 4