**Phase 1: Problem identification:**



A well-known company dedicated to the development of games, know for it´s realism in physics and real life facts has been hired by the NASA S.A.S to develop a software capable of simulating with precision space travels. The main objective of the software is to calculate the shortest and most efficient routes for interplanetary, intergalactic and special travels. The idea is to avoid material, financial and human loss. However, most of the travels will be made only through simulations due to there aren’t enough resources nor technology to execute them in real life.

The company plans to develop the software as a videogame that will also have the purpose of gathering funds and continue with investigations in outer space. In the video game travels will be simulated using different means of transportation, including worm holes and black holes. However, the purpose of the videogame is to reach the **habitable planet** doesn't matter where the ship is, Earth or any point of interest.

The company also has the intention of including the possibility of arriving or landing on other point of interest different from the habitable planet; this is in case the crew find something interesting or if they need supplies. Furthermore, the intention is to look for the shortest and most efficient route to get to a destination.

**Requerimientos:**

| **Clients** | NASA S.A.S. |
| --- | --- |
| **User** | NASA and players |
| **Funcitonal Requirements** | The system must allow:  R1: Show the shortest path between Earth and the habitable planet.  R2: Verify the possibility to reach the habitable planet from a specific point.  R3: To navigate from a specific point to another point of interest. |
| **Context** | A videogame development company, known for its realism, has been hired by NASA S.A.S. to create a software that simulates space travels with precision. The objective is to calculate efficient and safe routes for interplanetary, intergalactic and space travels, avoiding material and human losses.  The software will be developed as a videogame to gather funds. Also, in case the crew need supplies or if they decide to land on a point different from the habitable planet they will be able to do it. |
| **Process requierements:** | ➔ R1: The software has to be developed in a group of 3 developers.  ➔ R2. The system has to be uploaded to the Github version control system and to be kept private until release day. |
| **Non-functional requirements** | ● R1: The user interface has to be simple and easy to understand so that the users feel comfortable with it.  ● R2. The software has to be developed in java. |

| Name | R1. The shortest path between Earth and the habitable planet. | | |
| --- | --- | --- | --- |
| Summary | The system has to find and print which is the shortest path between Earth and the habitable planet and they could use a worm holes or a black holes. | | |
| Input | Name of input | Data type | Selection of condition or repetition |
| interestPoint | Vertex |  |
| General or necessary activities to obtain the result | The system receives a data base that contains vertices. Once uploaded to the system, the software has to use the Dijkstra algorithm to find the most efficient path and print the way to get there that was calculated. | | |
| Result or postcondition | The most efficient path is obtained to get to the destination. | | |
| Output | Output name | Date type | Selection of condition or repetition |
| message | String |  |

| Name | R2. Verify the possibility to reach the habitable planet from a specific point. | | |
| --- | --- | --- | --- |
| Summary | The system has to find and print what is the shortest path from a location to the other interest point and will be able to use worm holes or black holes. | | |
| Input | Input name | Data type | Selection of condition or repetition |
|  |  |  |
| General or necessary activities to obtain the result | The system has to calculate and demonstrate if it is possible and how. | | |
| Result or postcondition | The most efficient path is obtained to get to the interest point or not, depending on the initial location. | | |
| Output | Output name | Data type | Selection of condition or repetition |
| message | String |  |

| Name | R3. To navigate from a specific point to another point of interest. | | |
| --- | --- | --- | --- |
| Summary | The system has to find and print which is the shortest path between the inicial position and the interest point selected by the user | | |
| Input | Input name | Data type | Selection of condition or repetition |
| interestPoint | Vertex | Si el punto de interés no existe. |
| General or necessary activities to obtain the result | The system has to calculate and demonstrate if it is possible and how. | | |
| Result or postcondition | The most efficient path is obtained to get to the destination. | | |
| Output | Output name | Data type | Selection of condition or repetition |
| message | String |  |

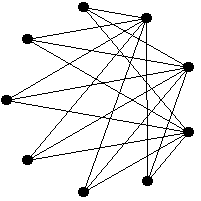
**Phase 2: Information gathering:**

To resolve this problem we’ll use the graph theorem, but to understand how the solution will develop, its necessary to understand the theorem with its BFS and Dijkstra algorythms. Therefore:

**¿What is it and how does a graph work?**

It’s a discrete structure composed by edges and vertices. The vertices will be stars, worm holes and black holes. The edges will be the journeys the ship will make.

Graph can also be represented in the following way:



Likewise, there are different types of graphs, in spite of that our solution will be as follows:

* Directed weighted multigraph:

This graph is capable of having many paths to get to the same vertex, nevertheless every path is conditioned to a specific direction.

**¿What do BFS and Dijkstra algorythms do?**

The BFS algorithm searches an element from our graph, and it does it in a systematic way visiting all the neighbors from the starting node before going to the neighbors of the first neighbor that was visited, and so on. This algorithm finds and shows the user the shortest path betweem planet Earth and a point of interest.

Dijkstra algorithm,for its part, works by finding the shortest path between the initial vertex and all the nodes of the graphs. In our problem this translates into finding the shortest path from planet Earth until the point of interest.

**Glossary:**

**BFS:**

Breadth-First Search (Búsqueda en anchura)

Ref: <https://www.educative.io/answers/what-is-breadth-first-search>

**Dijkstra:**

El algoritmo recibe el nombre de su creador, el científico informático holandés Edsger Dijkstra.

Ref: <https://www.analyticssteps.com/blogs/dijkstras-algorithm-shortest-path-algorithm>

References:

Menéndez Velázquez, A. (1998). Una breve introducción a la teoría de grafos. *Suma*.

**Phase 3: In search of creative solutions:**

**Alternative 1:**   
Our first idea is to use Dijkstra and BFS algorythms in a simple weighted graph. In our context the vertices are represented by stations that takes us to the planet destination which is our point of interest from an origin that is planet Earth.

A weight is assigned to the edges that connect the vertices which is expressed as a distance.

Dijkstra algorithm works finding the shortest path between stations selecting the smallest weight or distance. The path starts form planet Earth and ignoring the edges that have a bigger distance. Th stations connected to the current station are being visited and selects the smallest distance until the last station is visited. BFS algorithm selects the shortest path visiting every station but in this case the edges don’t have a numbers representing distances.

**Alternative 2:**

Algorythms Prim and Kruscal with a weighted simple graph. Prim algorithm will create a minimum spanning tree in which it connects with all the stations selecting the smallest distances that join the stations without creating any loops.

The algorithm starts with an initial station and then it adds the paths with less distance until the destination is reached.

Kruscal is an algorithm that finds the most optimum option in each phase instead of focusing on the global optimum. First, it organizes the distances from the smallest to the biggest. Then, the path with the smallest distance is added to the spanning tree and if that route creates a loop, then it´s rejected. After, more routes are added until all stations are reached creating a minimum spanning tree.

**Alternative 3:**

Algorythms Floyd Warshall and DFS with a simple weighted graph. The algorithm Floyd Warshall is created with a matrix which contains the stations in pairs as rows and columns. The current distance is updated if the next “ij” have a distance that is smaller than the current one.

The DFS algorithm wouldn’t make this alternative to work completely due to it doesn’t find the fastest route between stations and the planet. It only shows a path that gets to the planet going deep in the branch of stations showing all possible paths.

**Phase 4: Transition from Ideas to Preliminary Designs.**

To our consideration there aren’t any ideas that are not viable since every each one of them models the problem correctly according to the context. More information following bellow on the ideas that were not discarded.

**Alternative 1**

Using Dijkstra and BFS allows us to select the shortest path ignoring the longest distances. With Dijkstra the algorithm chooses the edge that has the smallest weight finding the shortest path to the destination.

BFS doesn’t use any weighted graph so there aren´t any distances it visits each level and all its neighbors and by recording the parents it can trace back the shortest path.

**Alternative 2**

Prime and Kruscal are also viable since it finds the shortest path. Prim algorithm finds the shortest path without creating any cycles. There’s an initial vertex and then the edges are added until the point of interest is reached.

Kruscal organizes the distances from the smallest to the biggest. Then the smallest distance is added to the spanning tree. The route cannot male a loop. If al vertexes are reached then they are also added to the spanning tree.

**Alternative 3**

Although DFS doesn’t go for the shortest path it explores all vertices and corresponding edges length showing all possible paths. Floyd Warshall is created with a matrix containing each vertex as a coordinate row and column. The coordinate is updated if there is a distance that is smaller than the current one.

**Step 5: Evaluation and Selection of the Best Solution:**

**Criteria:**

The criteria that will allow evaluating the alternative solutions must be defined and based on this result choose the solution that best meets the needs of the problem. The criteria we chose in this case are the ones we list below. Next to each one a numerical value has been established with the aim of establishing a weight that indicates which of the possible values of each criterion have the most weight (i.e., they are more desirable).

***- Criteria 1:***  
 Uses distances according to the context.

o [2] Yes

o [1] No

***- Criteria 2:***Less quantity of vertexes.

o [3] Has the fewest vertices

o [2] Has more vertices than expected

o [1] Has too many vertices

***- Criteria 3:***

Temporal complexity.

o [2] Fast

o [1] Slower

***- Criteria 4:***   
Coding efficiency.

o [2] Yes

o [1] No

|  | Criteria 1 | Criteria 2 | Criteria 3 | Criteria 4 | Total |
| --- | --- | --- | --- | --- | --- |
| Alternative 1 | 2 | 3 | 2 | 2 | 9 |
| Alternative 2 | 2 | 2 | 1 | 1 | 6 |
| Alternative 3 | 2 | 1 | 1 | 1 | 5 |

*Selection:*

According to the previous evaluation, Alternative 1 should be selected since it obtained the highest score according to defined criteria.