**Fase 1: identificación del problema:**



Una reconocida empresa dedicada al desarrollo de videojuegos, conocida por su realismo en las físicas y hechos de la vida real, ha sido contratada por la NASA S.A.S. para desarrollar un software capaz de simular con precisión los viajes espaciales. El objetivo principal del software es calcular las rutas más cortas y eficientes para los viajes interplanetarios, intergalácticos y espaciales, evitando pérdidas materiales, monetarias y humanas. Sin embargo, se debe tener en cuenta que muchos de estos viajes solo serán realizados en simulaciones, ya que no se cuenta con los recursos ni la tecnología necesarios para llevarlos a cabo en la realidad.

La empresa planea desarrollar el software como un videojuego que también servirá para recaudar fondos y continuar con las investigaciones en el campo espacial. En el videojuego, se podrán simular los viajes utilizando diferentes medios de transporte, incluyendo agujeros de gusano y agujeros negros, aunque la NASA S.A.S. considera que estos métodos son extremadamente peligrosos e imposibles en la realidad. Por lo tanto, el uso de estos "medios de viaje" será una opción adicional en el juego para agregar emoción, pero no se considerarán parte de la realidad del software.

La empresa también tiene la intención de incluir la posibilidad de regresar de un punto dado al planeta Tierra o a una estación espacial internacional en caso de que ocurran problemas o fallas con la tripulación o la nave. Además, se busca calcular la ruta más corta y eficiente para llegar a un destino sin utilizar agujeros de gusano ni agujeros negros, para mantener la autenticidad y realismo del juego según los estándares establecidos por la NASA S.A.S.

**Requerimientos:**

|  |  |
| --- | --- |
| **Cliente** | NASA S.A.S. |
| **Usuario** | NASA and players |
| **Requerimientos funcionales** | El sistema debe permitir:  **R1**: Encontrar y mostrar al usuario el camino más corto entre la tierra y un punto de interés.  **R2**: Verificar la posibilidad de regreso a la Tierra  **R3**: Mostrar el camino más eficiente entre la tierra y un punto de interés sin usar agujeros de gusano o agujeros negros. |
| **Contexto** | Una empresa de desarrollo de videojuegos, reconocida por su realismo, ha sido contratada por la NASA S.A.S. para crear un software que simule viajes espaciales de forma precisa. El objetivo es calcular rutas eficientes y seguras para los viajes interplanetarios, intergalácticos y espaciales, evitando pérdidas materiales y humanas. El software se desarrollará como un videojuego para recaudar fondos y se enfocará en la realidad, excluyendo agujeros de gusano y agujeros negros como medios de transporte. También se contempla la opción de retorno a la Tierra o a una estación espacial en caso de problemas. |
| **Requerimiento de procesos:** | ➔ R1: El programa debe ser desarrollado en un grupo de 3 desarrolladores  ➔ R2. El sistema debe ser subido al sistema de control de versiones GitHub y mantenerse privado hasta el día del lanzamiento. |
| **Requerimientos no funcionales** | ● R1: La interfaz de la aplicación debe ser simple y fácil de entender para que los usuarios se sientan cómodos con ella.  ● R2. El programa debe ser desarrollado en Java. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nambre | R1. Camino más corto entre la tierra y un punto de interés. | | |
| Resumen | El sistema debe encontrar e imprimir cual es el camino más corto entre la tierra y un punto de interés elegido por el usuario y podrá usar agujeros de gusano o agujeros negros | | |
| Entradas | Nombre de entrada | tipo de dato | Selección de condición o repetición |
| interestPoint | Vertex | Si el punto de interés no existe. |
| Actividades generales o necesarias para obtener el resultado | El sistema recibe una base de datos que contenga las aristas y vértices, una vez cargado en el sistema y después de haber recibido el punto de interés, el programa debe utilizar el algoritmo de Dijkstra para encontrar el camino más eficiente e imprimir la manera de llegar hallada. | | |
| Resultado o poscondición | Se obtiene el camino más eficiente para llegar al destino. | | |
| Salidas | Nombre de salida | tipo de dato | Selección de condición o repetición |
| message | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Nambre | R2. Verificar la posibilidad de regreso a la Tierra. | | |
| Resumen | El sistema debe encontrar e imprimir cual es el camino más corto de un punto a la tierra o y podrá usar agujeros de gusano o agujeros negros | | |
| Entradas | Nombre de entrada | tipo de dato | Selección de condición o repetición |
|  |  |  |
| Actividades generales o necesarias para obtener el resultado | El sistema debe calcular y demostrar si es posible y como volver a la tierra partiendo del punto donde se encuentre. | | |
| Resultado o poscondición | Se obtiene el camino más eficiente para llegar de vuelta a la tierra o no, dependiendo de la ubicación inicial. | | |
| Salidas | Nombre de salida | tipo de dato | Selección de condición o repetición |
| message | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | R3. Mostrar el camino más eficiente entre la tierra y un punto de interés sin usar agujeros de gusano o agujeros negros. | | |
| Resumen | El sistema debe encontrar e imprimir cual es el camino más corto entre la tierra y un punto de interés elegido por el usuario, pero esta vez no podrá usar agujeros de gusano ni agujeros negros por apelación al realismo. | | |
| Entradas | Nombre de entrada | tipo de dato | Selección de condición o repetición |
| interestPoint | Vertex | Si el punto de interés no existe. |
| Actividades generales o necesarias para obtener el resultado | El sistema deberá usa Dijkstra condicionado por el hecho de no poder usar aquellos vertices que correspondan a los agujeros negros y agujeros de gusano | | |
| Resultado o poscondición | Se obtiene el camino más eficiente para llegar al destino. | | |
| Salidas | Nombre de salida | tipo de dato | Selección de condición o repetición |
| message | String |  |

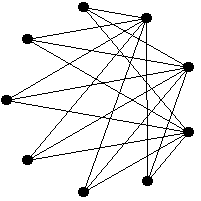
**Fase 2: Recolección de información:**

Para resolver este problema utilizaremos el teorema de grafos, pero para entender cómo se hará la solución es necesario comprender este teorema con sus algoritmos de BFS y Dijkstra, por lo tanto:

**¿Qué es y cómo funciona un grafo?**

Es una estructura discreta compuesta de aristas y vértices, en nuestro caso los vértices serán planetas, estaciones espaciales, estrellas agujeros negros y de gusano etc, mientras que las aristas serán los recorridos que puede hacer la nave.

Los grafos también suelen representarse de la siguiente forma:



Asimismo, existen diferentes tipos de grafos, pero para nuestra solución utilizaremos el siguiente:

* Multigrafo dirigido:

Este grafo permite tener varios caminos para llegar a un mismo vértice, pero así también cada camino está condicionado por una dirección específica.

**¿Qué hacen los algoritmos de BFS y Dijkstra?**

El algoritmo *BFS* busca un elemento de nuestro grafo y lo hace de manera sistemática visitando todos los vecinos del nodo por el cual partió antes de seguir con los vecinos del primer vecino que visitó y así sucesivamente, este algoritmo servirá para encontrar y mostrar al usuario el camino más corto entre la tierra y un punto de interés.  
El algoritmo *Dijkstra,* por su parte, sirve para hallar el camino más corto entre el vértice inicial y todos los nodos de los grafos, en nuestro problema, esto se traduce a hallar el camino más corto desde la tierra hasta el punto de interés.

**Glosario:**

**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>

Citas:

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

**Fase 3: Búsqueda de soluciones creativas:**

**Imagen de lluvia de ideas:**

**Alternativa 1:**

La primera idea sería utilizar los algoritmos de Dijkstra y BFS en un grafo simple dirigido ponderado que en nuestro contexto los vértices serían representados por las estaciones que llevan al planeta destino que sería el punto de interés desde un origen que sería la Tierra. A las aristas que conectan los vértices se les asigna un peso que se expresa como distancia.

El algoritmo Dijkstra funciona hallando el camino más corto entre las estaciones escogiendo el menor peso o distancia partiendo de la Tierra e ignorando las distancias que tengan una distancia mayor. Así mismo va visitando las estaciones que están conectados la estación actual y escoge la menor distancia hasta que haya visitado a la última estación. El BFS halla el camino mas corto descubriendo cada una de sus estaciones en una lista de adyacencia, pero en este caso las distancias no tienen números de longitudes.

**Alternativa 2:**

Algoritmos Prim y Kruscal con un grafo simple dirigido ponderado. El algoritmo Prim va a crear un árbol de expansión mínima en el cual se conecta por todas las estaciones escogiendo las distancias más pequeñas que unan esas estaciones sin crear bucles. El algoritmo empieza con una estación inicial y luego va agregando los caminos con menor distancia hasta que su destino haya sido alcanzado. El algoritmo Kruscal es un algoritmo que encuentra la opción más optima en cada fase en ves de enfocarse en el óptimo global. Primero se organizan las distancias de menor a mayor peso. Luego se agrega esa ruta que tiene el menor peso al árbol expansivo y si esta ruta crea un ciclo entonces se rechaza. Luego se continúa agregando las rutas hasta que se hayan alcanzado todas las estaciones creando un árbol mínimo expansivo.

**Alternativa 3:**

Algoritmos Floyd Warshall y DFS con un grafo simple dirigido ponderado. El algoritmo Floyd Warshall se crea mediante una matriz que es la que contiene las estaciones en parejas entre columnas y filas. La distancia actual se actualiza si el siguiente par de estaciones ij tienen una distancia mas costa que el actual.

El algoritmo DFS realmente harina que esta alternativa no funcione del todo ya que no encontrar la ruta mas rápida entre estaciones y el planeta ya que solo nos mostraría un camino que llegue al planeta de manera que profundiza en una rama de estaciones mostrando todos los caminos posibles.

**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.