UNIVERSIDAD ICESI

Tarea Integradora III

Facultad de Ingeniería

Algoritmos y Estructuras de Datos I Periodo de 2021

Carolina Pasuy Pinilla A00358427 , Juan Manuel Palta A00369611 ,

Anderson Cardenas A00361998

Programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad Icesi, Cali Colombia

Functional Requirements

R1. Find the minimum path on the Bikini Bottom map.

R2. Find the minimum path to collect all the ingredients.

R3. Find the minimum path to collect the keys with Mr. Krabs.

R4. Create and save all the information of a player.

R5. Generate the player's score according to the time it takes to find the minimum path on the map.

R6. Generate the maps of the clues.

R7. Generate the available clues and deliver the rewards if you hit the two available challenges.

R8. Show the rewards of the challenges only if the user answers correctly, otherwise nothing is shown.

R9. Show the scores of all users who have played.

Non-functional requirements

R1. Implement the Graph structure

R2. Create the Adjacency List and Adjacent Matrix representations for the graph structure and implement one.

R3. Analyze, design, and implement one of each:

- Routes on Graphs (BFS, DFS)
- Minimum Weight Roads (Dijkstra, Floyd-Warshall)
- Minimum Cover Tree -MST- (Prim, Kruskal)

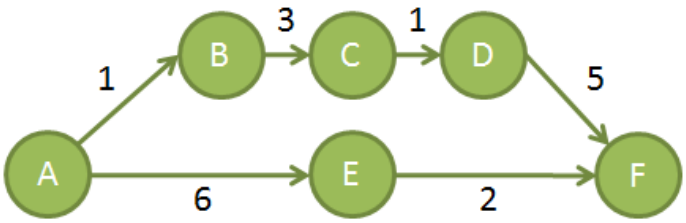
R4. Use binary search trees to save users.

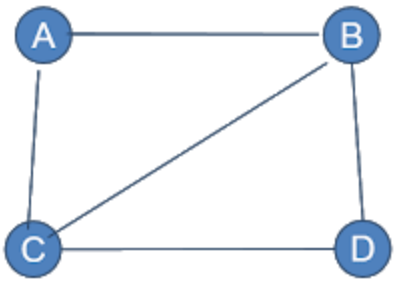
Engineering Method

1.Identification of the problem: SpongeBob needs to get to the Crabby Krab as fast as possible to prepare some good Kangre Burger. To do this, you have the Bikini Bottom map to be able to choose the minimum path. However, you have 3 clues that can help you figure this out. The first is the minimum road distance. The second and third are challenges in which you will have only one chance to unlock the rewards. In the second challenge Bob must collect all the ingredients in the shortest distance possible, if he succeeds, the reward will be the distance as he travels the main map. The third tries to go to Mr. Krabs for the keys to Crustacea Crabby in the shortest possible distance.

2.The collection of the necessary information:

Graph: A graph is a non-empty set of objects called vertices (or nodes) and a selection of pairs of vertices, called edges (edge in English) that can be oriented or not. Typically, a graph is represented through a series of points (the vertices) connected by lines (the edges). The Formally, a graph is defined as {\ displaystyle G = (V, A)} {\ displaystyle G = (V, A)}, where V is a set whose elements are the vertices of the graph and A is a set whose elements are the edges , which are pairs (ordered if the graph is directed) of elements in V.





Source:

http://www.unipamplona.edu.co/unipamplona/portalIG/home_23/recursos/general/11072012/grafo3.pdf

Adjacency List: In this data structure the idea is to associate to each vertex i of the graph a list

containing all those vertices j that are adjacent to it. In this way, it will only reserve memory for arcs adjacent to i and not for all possible arcs that could have i as origin. The graph, therefore, is represented by means of a vector of n components (if | V | = n) where each component is going to be an adjacency list corresponding to each of the vertices of the graph. Each element of the list consists of a field indicating the adjacent vertex.

Source:

https://es.slideshare.net/FrankDoria/lista-de-adyacencia

Adjacency Matrix: It is a Boolean matrix that represents the connections between pairs of vertices. The matrix ofadjacency of a graph is symmetric. If a vertex is isolated then the corresponding row(column) is composed only of zeros. If the graph is simple then the adjacency matrixit contains only zeros and ones (binary matrix) and the diagonal is made up of only zeros.

Source:

https://www.utm.mx/~mgarcia/ED4(Grafos).pdf

Depth-first Traversal (DFS): Is a technique that is used to traverse a tree or a graph. DFS technique starts with a root node and then traverses the adjacent nodes of the root node by going deeper into the graph. In the DFS technique, the nodes are traversed depth-wise until there are no more children to explore.

Once we reach the leaf node (no more child nodes), the DFS backtracks and starts with other nodes and carries out traversal in a similar manner. DFS technique uses a stack data structure to store the nodes that are being traversed.

Breadth-first Traversal (BFS): Is a technique uses a queue to store the nodes of the graph. As against the DFS technique, in BFS we traverse the graph breadth-wise. This means we traverse the graph level wise. When we explore all the vertices or nodes at one level we proceed to the next level.

Source:

https://www.softwaretestinghelp.com/java-graph-tutorial/

Dijkstra's algorithm: Also called minimum paths algorithm, it is an algorithm for determining the shortest path given a vertex origin to the rest of vertices in a graph with weights on each edge. Its name refers to Edsger Dijkstra, who first described it in 1959.

Some characteristics of the algorithm it is:

- A greedy algorithm.
- Work in stages, and take the best solution at each stage without considering future consequences.
- The optimum found in one stage can be modified later if a better solution emerges.

Source:

https://www.ecured.cu/Algoritmo_de_Dijkstra

Kruskal's algorithm: Kruskal's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which:

- Form a tree that includes every vertex.
- Has the minimum sum of weights among all the trees that can be formed from the graph.

It falls under a class of algorithms called greedy algorithms that find the local optimum in the hopes of finding a global optimum.

Source:

https://www.programiz.com/dsa/kruskal-algorithm

Prim's Algorithm: algorithm, Prim’s algorithm is also a greedy algorithm. It starts with an empty spanning tree. The idea is to maintain two sets of vertices. The first set contains the vertices already included in the MST, the other set contains the vertices not yet included. At every step, it considers all the edges that connect the two sets, and picks the minimum weight edge from these edges. After picking the edge, it moves the other endpoint of the edge to the set containing MST.

A group of edges that connects two set of vertices in a graph is called cut in graph theory. So, at every step of Prim’s algorithm, we find a cut (of two sets, one contains the vertices already

included in MST and other contains rest of the vertices), pick the minimum weight edge from the cut and include this vertex to MST Set (the set that contains already included vertices).

Source:

https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/

Floyd Warshall Algorithm: the solution matrix same as the input graph matrix as a first step. Then we update the solution matrix by considering all vertices as an intermediate vertex. The idea is to one by one pick all vertices and updates all shortest paths which include the picked vertex as an intermediate vertex in the shortest path. When we pick vertex number k as an intermediate vertex, we already have considered vertices {0, 1, 2, .. k-1} as intermediate vertices. For every pair (i, j) of the source and destination vertices respectively, there are two possible cases.

1) k is not an intermediate vertex in shortest path from i to j. We keep the value of dist[i][j] as it is.

2) k is an intermediate vertex in shortest path from i to j. We update the value of dist[i][j] as dist[i][k] + dist[k][j] if dist[i][j] > dist[i][k] + dist[k][j]

The following figure shows the above optimal substructure property in the all-pairs shortest path problem.

Source:

https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/

3.Search of creative solutions: All ideas raised at this point were done using the brainstorming method.

- A game that contains three mini-games, where each one implements the three algorithms required by the statement. One of the minigames was going to be a maze where he had to find the minimum path to find the exit with Dijkstra algorithm. Another was going to resort to being a clue game where he had to find a treasure with all the clues, for there the Prime algorithm would be implemented. The last minigame would deal with a puzzle where each of the pieces was traversed with the DFS.
- A game that had as a theme Sponge Bob and it was about finding the minimum path to be able to go from his house to the Krabi Crustacean using Dijkstra algorithm., and putting 3 tracks, two where different maps were used, and there the algorithm was applied Kruskal for one and the Bfs for the other.

- A game that had a cave as its theme. The game would be about finding the minimum path of a cave by choosing which room to enter, and the luck factor would define the distance between each room. For this, the Dijkstra algorithm would be used for the minimum path and the DFS to show the entire map of the cave.
- An application on a European airline, where all the destinations available to travel with the BFS will be shown. With the Floyd-Marshall you will find the minimum path that you would have to take from one destination to another so that the trip would cost as little as possible and with the Kruskal algorithm you will find all the destinations you would have to take to be able to take a star flight all over the continent.
- A game about Jumanji, the map of the game board brings many alternatives on the way, but to quickly end the nightmare, the player is asked to pass the game in the minimum path of the map, for this would be used Floyd -Marshall where the player can observe the available distances and choose which would be the best option. You will also be presented with all the nightmares that will happen to you with the DFS algorithm and to finish a quiz of questions using the Prime algorithm.

- An application similar to that of Pokémon's Go, but it would have a theme similar to that of Anime, where the various characters would be collected from a map that shows you the minimum path that there is from where you are to each character implementing the Dijkstra, and it would show you the total of objectives that you have with the BFS, in addition to this it would show you the path that you would have to travel to be able to obtain all the characters with the Kruskal.

4.Moving from ideas to preliminary designs:

Mini Games: We discard the first idea of the mini games since we are not efficient in terms of the personal objectives of the game since we want to implement either a game or an application that is complete, where all the requirements are solved in a single program and not in one program but with different components. In addition, it would take more work to implement the logic of 3 different games and graphics.

Cave-Race: We discarded this idea since it did not meet the general objectives of the project, because we did not know how to implement any coating tree algorithm, so it would be incomplete, in addition to the fact that the luck factor should not be in a game, everything should comply with a logic already proposed and in terms of the graphic part it was going to be complicated to be able to make the game so that two players could play it at the same time.

Anime: We discarded this idea since it seems to us that it would require much more time to implement it since new tools would have to be used to get maps that are shown in real state, which we have not seen and it would require much more work to learn and do.

The other ideas seem appropriate to us to make the respective evaluations and choose the best one, although first we add more information to each one:

Alternative 1:

Jumanji: The map in which the game takes place would be a directed graph where its vertices would be the obstacles that will have to pass, and it would be unweighted.

Alternative 2:

European Airline: The map of this airline will be a weighted graph where the weight of the edges is the cost of passage between each country, and it will be directed. The countries would be the vertices.

Alternative 3:

Bob Sponge Game: The game would have three maps. The main one that would be the Bikini Bottom, a weighted and undirected graph. The second would be a map of Bob's friends, an unweighted and undirected graph, where he will have to go to Mr. Krabs to collect some keys, and the last would be the ingredients of a Kangre Burger, a weighted and undirected graph.

5. Evaluation and selection of the best solution:

Criterias:

- **Solution precision:** The solution is viable and meets the effective development of all the requirements of the problem**.**

    1. Imprecise
    2. Almost Accurate
    3. Accurate

- **Effectiveness:** Regardless of the complexity, the solution complies with a perfect effectiveness lacking any logical error.

    1. No Effective
    2. Effective Medium
    3. Effective

- **Usability:** The solution can be used because it meets all the proposed objectives.

1. Not usable at all

2. Complex

3. Usable

- **Accessible: The program presents easy accessibility to the game's functionalities.**

1. Not accessible

2. Moderately accessible

3. Accessible

Evaluating the previous criteria in the alternatives that remain, we obtain the following table:

	Solution Precision	**Effectiveness**	**Usability**	**Accessible**	Total
Alternative 1	2	3	3	3	11
Alternative 2	3	1	3	2	9
Alternative 3	3	3	3	3	12

Therefore, following the previous evaluation, it is agreed to work on Alternative 3, since it contains all the requirements raised in the statement, encompasses all the necessary and unnecessary methods in the data structure used and is a creative idea to implement.