

ID: 12859191938

Name: Arda Tarım

Section: 1

Assignment number: 1

## Problem Statement and Code Design

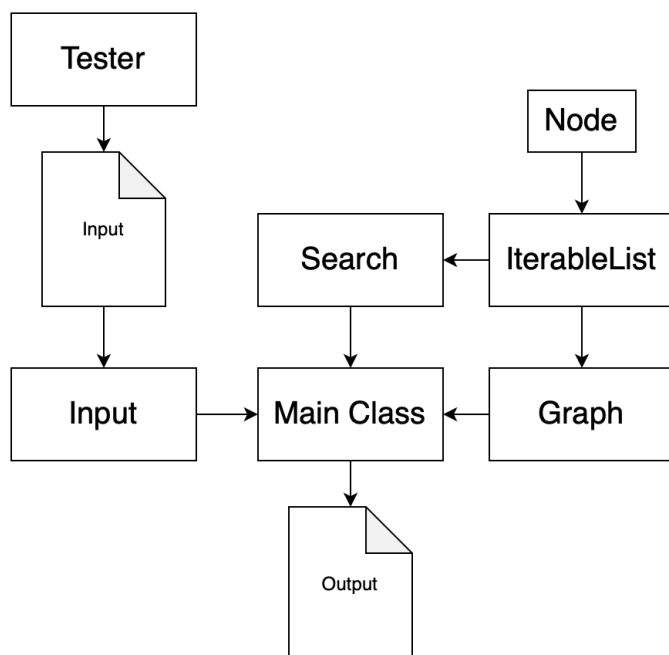
There are two similar questions given in the assignment, and I need to implement a Graph structure to solve them.

Firstly, I implemented a basic Graph class. I referenced the undirected Graph API from the book. I created a data structure called IterableList for the Graph's adjacent list. This class is basically a generic LinkedList/Queue that can return an Iterable. Also, I created the appropriate Node class for my IterableList.

Then, I created the Search class to find paths in the Graph. Search class is a basic BreadthFirst Search algorithm implementation and uses IterableList as the queue.

I created the Input class to read data from .txt files or input stream. Then I created an input algorithm to read the input data for the given questions. This data to create graphs and solve the questions.

Finally, I created the Tester class to create tests for the algorithms. The tests will be used to find where algorithms fail.



This is the basic schematic of my classes.

## Implementation and Functionality

### Main Class for Question1:

Firstly, Input and Graph classes are initialized. Then the data is read by the input class for Graph to be created.

BFS class is initialized and shortest path from start to end vertex is calculated with the findPath() method. The path information is recored to a String for printing.

After that 2 arrays are created to calculate the total time needed to travel. Calculating the total time is very simple when we can visualise the problem. Here is the logic of calculating the total time.

- Green squares mean the ports are running and represented by 1's in the array.
- Red squares mean the ports are off and represented by 0's in the array
- Black squares represent the travel time.
- Travel can only occur if the ports are running(if 1 is in the current array index)

Example for 3 situations:

- 1- (statechange= 3mins, travel= 5mins) and total time is 11 minutes.
- 2- (statechange= 2 mins, travel = 2mins) and total time is 10 minutes.
- 3- (statechange= 3 mins, trave= 2mins) and total time is 16 minutes.



### Main Class for Question2:

Firstly, Input and Graph classes are initialized. Then the data is read by the input class for Graph to be created.

BFS class is initialized and shortest path from startVertex to includeVertex is calculated with the findPath method. The path information is recored to an IterableList. Then shortest path from includeVertex to startVertex is calculated with the findPathExclude method using the firstPath as excluded vertices.

Lastly, both paths are combined into one. Then that path is copied into an array, gets sorted in lexicographic order to be printed.

### Graph:

Graph class implements basic undirected Graph API. Main functions in the class are Constructor(), addEdge(a,b) and adj(a). Graph can be initialized with a .txt file or from an input stream. addEdge() method adds an edge between the vertices a and b. Adj() method returns an IterableList containing all the neighbour vertices.

### IterableList:

IterableList is a custom generic class implementing the Iterator interface. The class is created for this assignment. It can be used as a Linked List or a Queue and is able to return an Iterator. So it is called an IterableList. It includes basic functions like add(), poll(), isEmpty() and has additional methods like getSize(), contains() and reverse().

### Node:

Node class is the building block of IterableList. It is a generic class and holds two variables data<T> and reference to the next Node.

### Search:

Search class is the implementation of Breadth First Search. It can be initialized with a Graph and has two methods called findPath and findPathExclude.

findPath() method initialized 3 arrays marked, edgeTo and dist. Then the start vertex is added to the queue. Until there are no vertices left in the queue the current vertex is added to the bfsOrder and adjacent vertices are checked. If the vertices are not visited (marked=false) they are added to the queue.

After this process, the search the algorithm starts from endVertex and travels to the startVertex. Recording the vertices it has visited to a list. Then the list is reversed to get the correct order from start to finish.

findPathExclude() method is the same as the previous one but takes one more parameter (IterableList exclude). Before the search, this method initializes a new graph and removes the vertices in the exclude list. (Visited vertices in the scope of question2). Then the same path algorithm is applied on the new graph.

### Input:

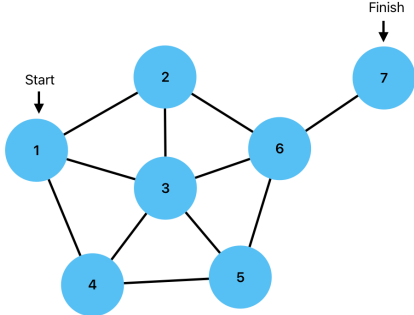
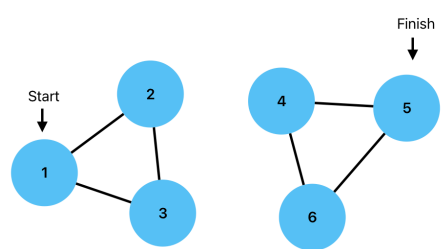

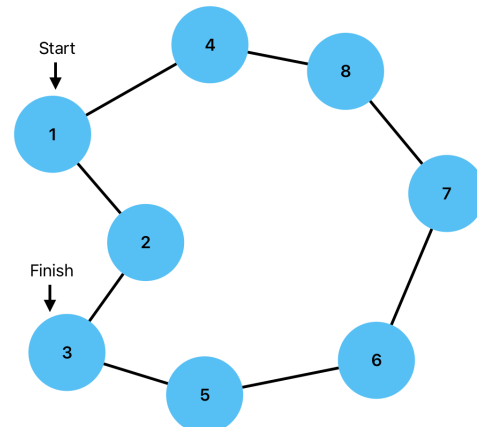
This class is can be used to create graphs from input streams. It can read .txt files or command line to get input. The constructor can be initialized with "filepath" for .txt files. The method readInt() is used to get the data from Input class.

## Tester:

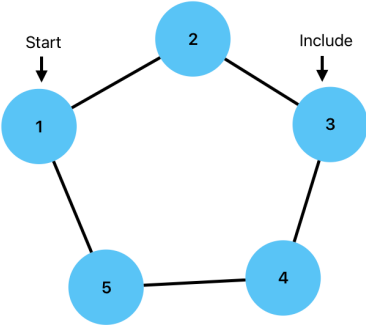
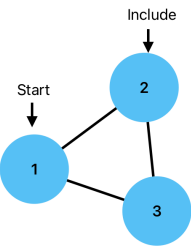

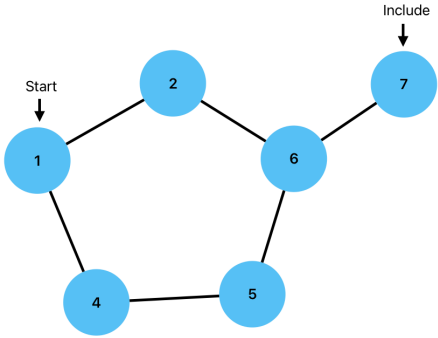
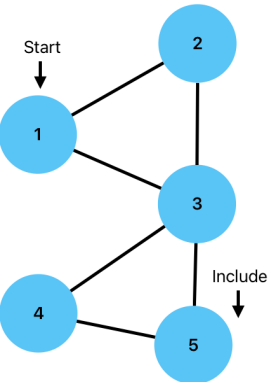
This class creates additional test cases to test the algorithms. The created test are .txt files which includes both the graph and information about the questions.

## Testing

For question1.

	4 1 2 6 7 17  Expected output.
	Exception in thread "main" java.lang.OutOfMemoryError: Java heap space  When there is no connection between start and end vertices.
	1 1 0  Expected output.
	3 1 2 3 11  Expected output.

For question2.

	<p>1 2 3 4 5</p> <p>Expected output.</p>
	<p>1 2 3</p> <p>Expected output.</p>
<p>Start / Include</p> 	<p>1</p> <p>Expected output.</p>
	<p>Exception in thread "main" java.lang.OutOfMemoryError: Java heap space</p> <p>When there is no valid path in the graph.</p>
	<p>1 2 3 4 5</p> <p>Expected output.</p>

## **Final Assessments**

What were the trouble points in completing this assignment?

While preparing the assignment i found it hard to add comments and explanations to my code.

What did you like about the assignment?

I liked the first question's time calculation. I think it makes the question more exciting in general.

What did you learn from it?

I practiced the graph structure and breadth first search.

## **References**

Algorithms, FOURTH EDITION, Robert Sedgewick and Kevin Wayne

[geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph](https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph)