**CPE113 Project**

Team : Carry Team

Members : Tulathorn Sripongpankul (58070503412)

Aiyarat Inthasuk (58070503437)

Oras Phongpanangam (58070503439)

Topic : Designing a new bus route.

Problem Summary

Given a set of locations, with information on the distance and the travel time between each pair of locations, find the best bus route (sequence of stops), where "best" is defined as the minimum travel time, minimum distance, or a weighted combination of both time and distance.

Then given a start time and a frequency (how often the bus runs), generate a schedule for the bus route (what time it will arrive at each stop, throughout the day).

Requirements list

* Program reads input file correctly.
* Program can support infinite number of stops
* Program creates 2D arrays containing information on distance or time travelled between 2 stations correctly.
* Program will calculate the shortest route to travel to all the stops, based on distance or time.
* Program can create a time table of each bus stop and save to text file.
* Program allows user to browse stations
* Program allows user to view/add/edit/delete stations.
* Program can save generated route for later viewing.
* The primary processing functions and data structures for this system must be written in standard C.
* This system must compile and run (correctly) on both Windows and Linux.
* The user interface for this system may be written in either C or Java.
* This system must use at least two of the data structure categories studied in CPE 113
* This system must be completed and working correctly by 17 May 2016.

Use Cases

Use case name: **Loading data from files.**

Actor: Any system user.

Goal: Downloading data file to create a database.

Main success narrative:

1. System found data files.
2. Successful download data files to program.
3. System make a database.

Alternative narrative1:

1. System cannot find data files.
2. System show a warning message.
3. Exit program.

Alternative narrative2:

1. System cannot download data files to program.
2. System show a warning message.
3. Exit program.

Use case name: **Create shortest distance bus route.**

Actor: Any system user.

Goal: To create the shortest distance bus route.

Main success narrative:

1. System download database.
2. Calculate the shortest distance route.
3. Save the shortest route.

Alternative narrative1:

1. System failed to access database.
2. Show the warning message.
3. Show the suggest message to remake database

Use case name: **Create shortest time bus route.**

Actor: Any system user.

Goal: To create the shortest time bus route.

Main success narrative:

1. System download database.
2. Calculate the shortest distance route.
3. Save the shortest route.

Alternative narrative1:

1. System failed to access database.
2. Show the warning message.
3. Show the suggest message to remake database

Use case name: **Add new bus stop.**

Actor: Any system user.

Goal: To add the new bus stop.

Main success narrative:

1. System access database.
2. System asked the distance betweent each bus stops.
3. System modifly new database.

Alternative narrative1:

1. System cannot access database.
2. System show the warning message.

Alternative narrative2:

1. System cannot modifly new databease
2. System show warning message
3. System deleate database

Use case name: **View route.**

Actor: Any system user.

Goal: To add the new bus stop.

Main success narrative:

1. System access database
2. System display bus route

Alternative narrative1:

1. System cannot access database.
2. System show the warning message.

Use case name: **Edit bus route.**

Actor: Any system user.

Goal: To edit the new bus route.

Main success narrative:

1. System access database
2. System asked which bus stop need to edit
3. Edit the bus stop
4. System modify database

Alternative narrative1:

1. System cannot access database.
2. System show the warning message.

Alternative narrative2:

1. System cannot find the bus stop.
2. System show the warning message.
3. System asked which bus stop need to edit again.

Alternative narrative3:

1. System cannot modifly new databease
2. System show warning message
3. System deleate database

Use case name: **Delete database.**

Actor: Any system user.

Goal: To remove database route.

Main success narrative:

1. System access database.
2. System delete database.

Alternative narrative1:

1. System cannot access database.
2. System show the warning message.

Use case name: **Create time table.**

Actor: Any system user.

Goal: To create the time table.

Main success narrative:

1. System access the generate route
2. System ask for start time and frequency
3. System generate time table
4. System write timetable files

Alternative narrative1:

1. System failed to access the route
2. System show the warning message
3. System exit the function

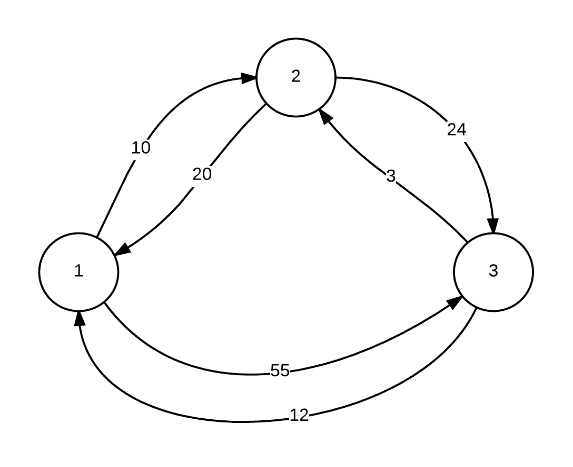
Alternative narrative2:

1. System failed to create a file
2. System show the warning message

Data Structure

* cost matrix
  + 2D array containing information on distance or time required in travelling between two stops

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 | 2 | 3 |
| 1 | - | 10 | 55 |
| 2 | 20 | - | 24 |
| 3 | 12 | 3 | - |

* + the first index is the stop to travel FROM and the second index is the stop to travel TO cost required in travelling from a to another stop can be different from the cost required in travelling from the other stop to the stop

* primary queue
  + data structure to help with routing algorithm
  + can conceptually hold any type of data, in this case, a “route”
  + similar to queue, but instead of “dequeuing” the first item in the queue, it will pop an item based on a set of specified rules. in this case,
    - 1. the most complete route will take priority, that is, the route with only 1 stop has low priority, while a complete route will be popped first
    - 2. for routes with same completeness, the route with lowest “lowerBound” will be popped first
  + (probably) will be implemented with heap
* route
  + structure used to store information of a route generated routing algorithm
  + the complete route will be of this type, but intermediate routes used in algorithm will be implemented this way to
  + fields:
    - int\* stopsIndex - pointer to array of index of stops in the route in order of arrival
    - int stopsCount - number of stops in this route
* stops information
  + store information of each stop apart from time and distance such as name, address, landmark, etc.
  + will use array of structures for simplicity, since searching is not required, and we will probably never have more than 100 entries due to the highly complex nature of the problem

Architecture Diagram

Priority Queue

Routing

User Interface

Stops Breaking

table

dbOperations

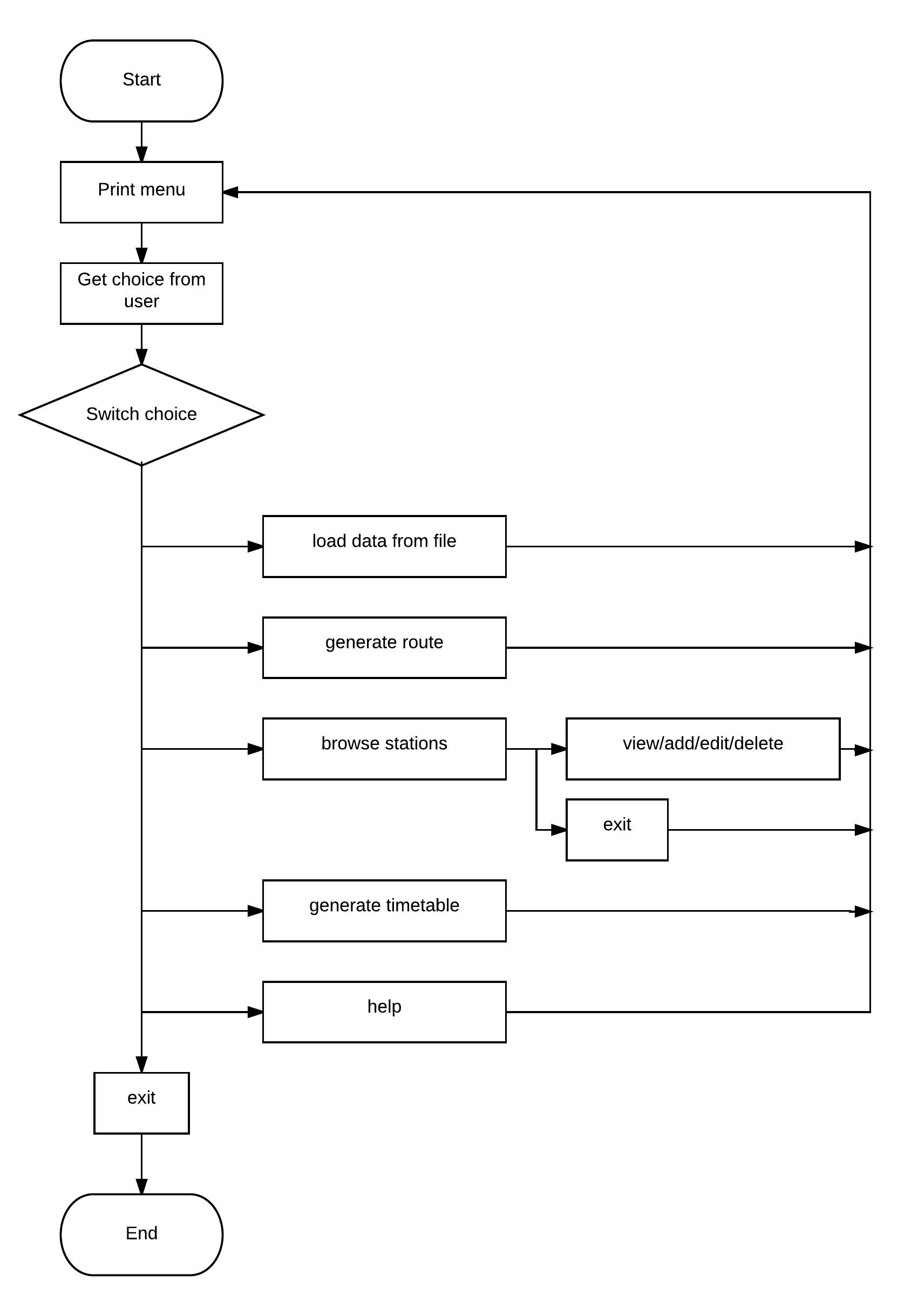
stopsDb

Architecture Explained

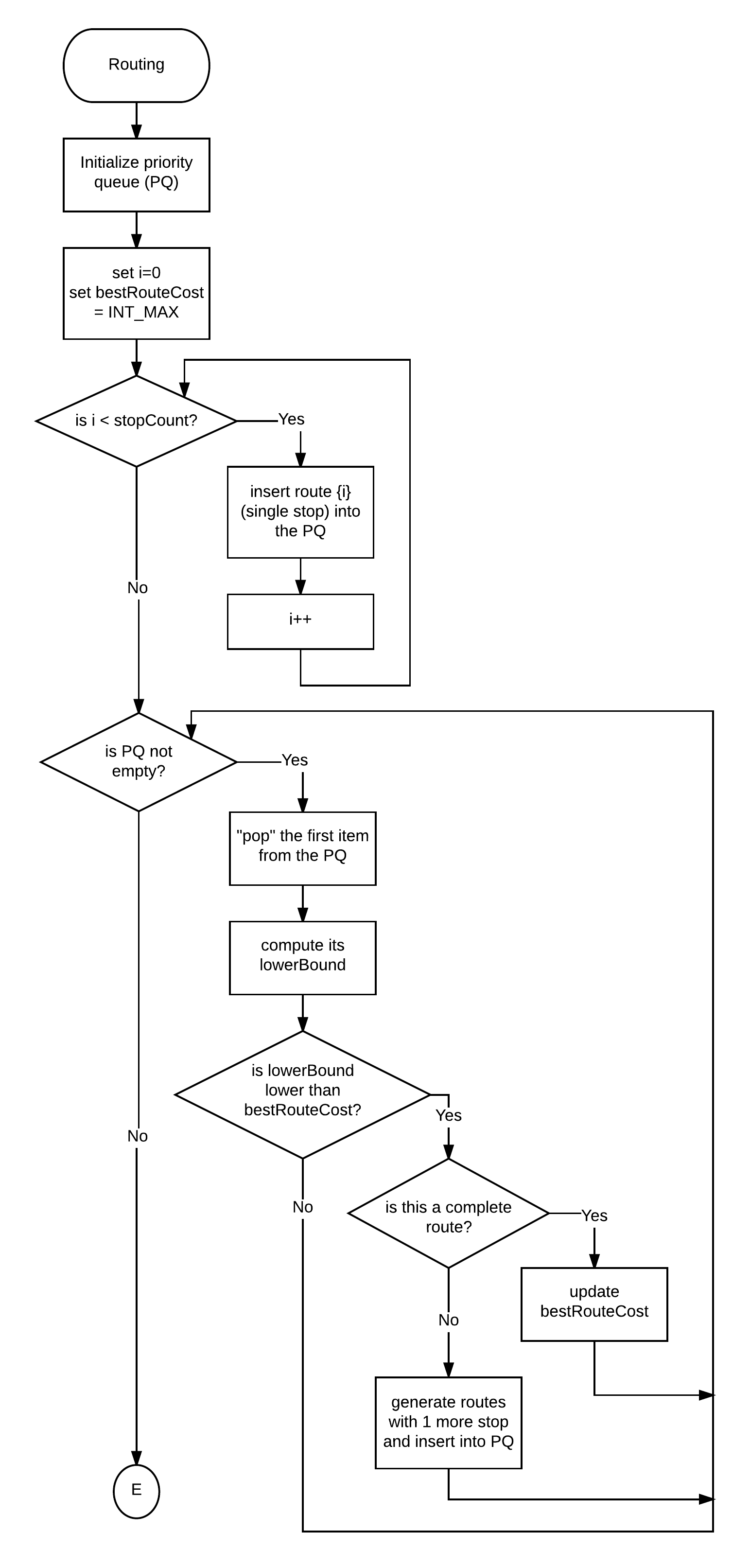
|  |  |
| --- | --- |
| User Interface | * Display menu * Control flow of user experience * Display errors * Get inputs |
| Routing | * Generate route from cost matrix |
| Table | * Generate table * Write table information to file |
| Stops Browsing | * View stop’s information * Add more stop * Delete a stop * Edit a stop’s information |
| Priority Queue | * Create new priority queue * Insert a new item into the queue * Calculate priority and sort accordingly |
| DB operations | * Read data from stops database (text file) * Convert data to correct data structure * Update information in database |

Flowchart

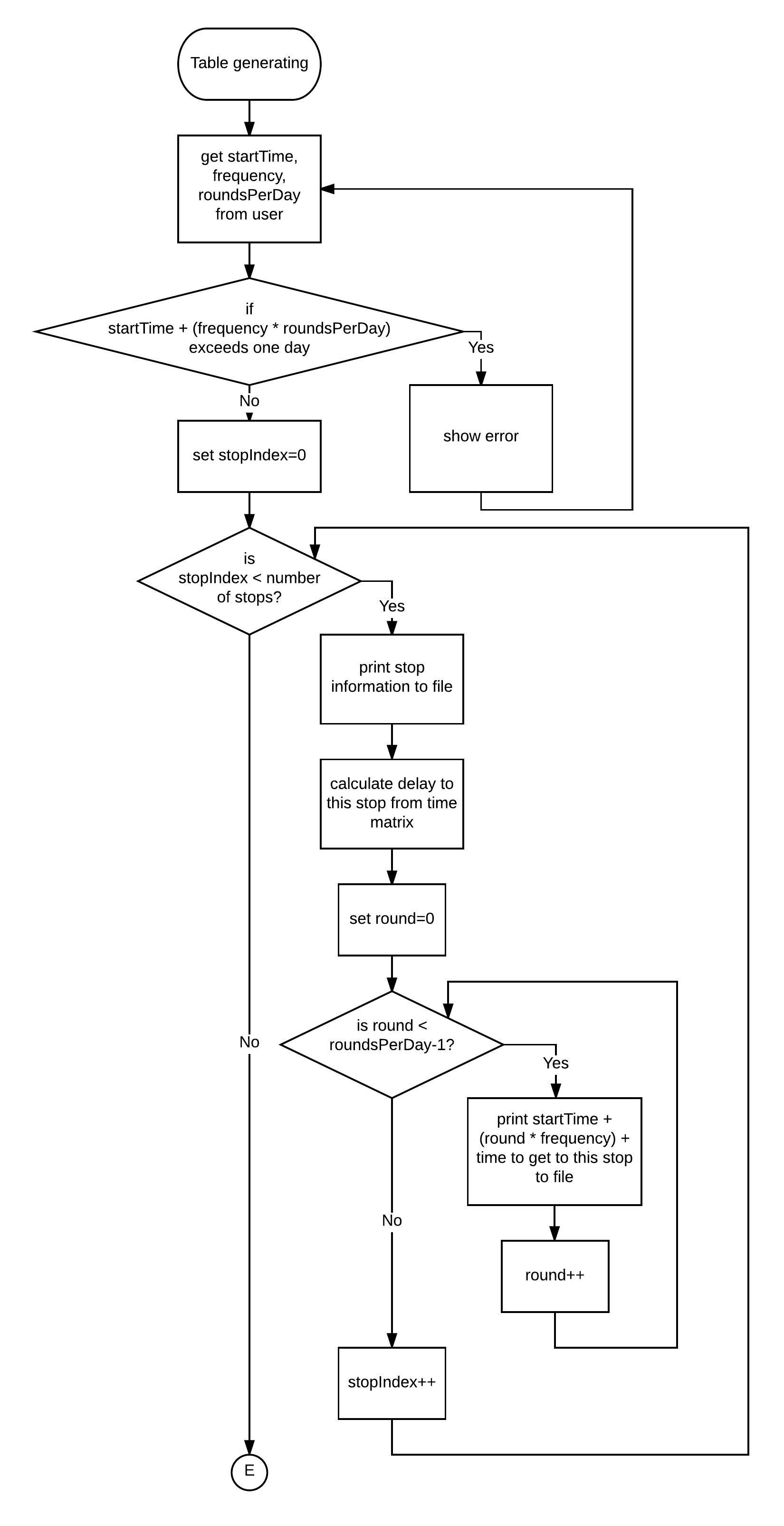
* Overview Flowchart



* Bus routing



* Table generating



Sample File Content

Stops database

4

Wat Phut, Phootabucha road

KMUTT, some road

Test, test

Test2, test2

0 10 25 26

23 0 35 55

12 55 0 22

32 5 2 0

0 23 34 45

23 0 432 384

34 324 0 239

8347 34 34 0

0,1,3,2

* First line tells the number of stops (in this case, 4)
* Next, set of lines about stops’ names, addresses, etc (all treated as strings separated by commas)
* lines for time matrix in minutes
* lines for distance in km
  + the dimension of matrices must be exactly n x n (n is number of stops)
  + the values in diagonal line are all 0
* last line for generated route
  + optional: it will be generated after routing

table output

Start time: 10:00 frequency: 10 minutes

Wat Phut, Phootabucha road

10:00

10:10

10:20

10:30

.

.

.

KMUTT, some road

10:10

10:20

10:30

10:40

.

.

.

* first line contains start time and frequency information
* next, name of stops, and arrival times, ordered by route