ISE 315 - Analysis of Algorithms
Assignment #2 Heapsort

Şeymanur Aktı
akti15@itu.edu.tr

Due: December, 22nd 23:55

Important Notes

- Please write your own codes, copying code parts from **books**, **websites** or any other sources **including your friends** is considered as **plagiarism** and will be treated seriously. You are expected to act according to the Code of Honor.
- Do not upload your codes to any public platform (e.g. Github) until the deadline of homework passes.
- Use C++/Python language and do not forget to compile your codes on Linux before submission.
- You may use STL but do not use built-in sorting functions.
- Do not forget to comment your code when needed. You **will be graded** for your comment quality.
- Submit your source codes and report files on Ninova before the deadline. Late submissions and submissions via e-mail may **not be accepted**.
- For any issues regarding the assignment, please contact Şeymanur Aktı (**akti15@itu.edu.tr**). Please send an e-mail if you have any questions.

I. Implementation (60 points)

You are given a dataset including the locations of 1.642 vehicles belong to the Take Me There company which offers transportation services in Istanbul. Hotel Bee is one of the business partners of Take Me There company. The hotel uses services of Take Me There for the transportation of guests staying at the hotel. Each morning, hotel concierge lists down the necessary information for the transportation of guests in the following files:

- **vehicles.txt** Initially available vehicles of Take Me There with following information:
  - **vehicle_ID**: Unique ID for vehicles
  - **location**: Location code where the vehicle is placed initially.
  - **distance**: Distance to Hotel Bee from the current location in kilometers.
  - **speed**: Average speed of the vehicle in km/h.
- **requests.txt** Transportation requests coming from the customers. First come first served, you should process the requests in the given order. This file includes following information.
  - **location:** Destination location.
  - **distance:** Distance of the destination location to the hotel.
  - **lucky_index:** Lucky number of the customer if any. This is used for **Task 2** below.

You are supposed to help the hotel concierge to arrange the information and call the vehicles efficiently, implementing the following tasks.

- **Task 1:** Build a minimum priority queue using heap structure keeping the vehicles in **vehicles.txt**. The key values should be the estimated **time** for the vehicles to reach to the Hotel Bee. You can simply calculate it by this formula:

$$\text{time} = \frac{\text{Distance to the hotel}}{\text{Speed of the vehicle}}$$

  We will use this heap for all of the following operations.

- **Task 2:** For each request in **requests.txt**, call the vehicle for the customer.
  - If the customer has no lucky numbers, the last column of the request will be 0. Then, call the vehicle which can arrive the hotel the fastest. **Extract** the called vehicle from the heap maintaining the heap property. Write down the ID of called vehicle into **call_history.txt**
  - If the customer has a lucky number written (**L**), the last column of the request will be different than 0. In this case, the hotel concierge must call the vehicle that positioned in the **L***th* index of the heap. Then, first **decrease** the key value of the vehicle in the related index to make it the first priority vehicle in the heap maintaining the heap property. Then, call this vehicle and **extract** it from the heap. Write down the ID of called vehicle into **call_history.txt**

- **Task 3:** After the called vehicle takes the customer to the destination, change the location and distance information of the vehicle as it will be at the destination place. Then, calculate the new key value (estimated time) and **insert** the vehicle into heap again.

Execution Details

Your code must run with one of these commands: **./a.out N** or **python3 file_name.py N**

Inputs:

- N is the total number of **extract, decrease** and **insert** operations that program should execute. During the runtime, count the number of these operations and exit the program when the N operations are executed.

Outputs:

- You should measure the running time for executing N operations (**extract, decrease** and **insert**) and a message including the elapsed execution time should be printed out on command line at the end. This only includes Task 2&3 operations, do not include the time for building the heap or I/O operations.
- **call_history.txt** file should contain the called vehicle IDs in order. One vehicle could be called multiple times as we insert the same vehicle into the heap again after calling it. **call_history_1000.txt** is included in the zip file as sample output for N = 1000.

Notice For calculating running time you can use clock() function under the ctime library. You can read more about it from here.

II. Report (40 points)

1. **(20 points)** Write down the corresponding heap operations for the **extract, decrease and insert** operations in the implementation. Then, give the asymptotic upper bounds for each operation and explain these bounds with your own words.
2. **(20 points)** For different values of N (1000, 10K, 20K, 50K, 100K) calculate the average time of multiple executions (at least 5 times). Report the average execution times in a table and prepare an Excel plot which shows the N – runtime relation of the algorithm. Comment on the results in detail considering the asymptotic bounds that you gave in Question 1.