Advanced Data Structures Project

Wayne Building Construction

Problem Statement:

Design an application which takes in different inserts of buildings and then constructs the different buildings based on their execution time. The application also involves frequent searches for different buildings to monitor the progress of construction of different buildings. It also has the efficiency of extracting of the building using the heap which orders its data based on its executed time. The minimum is present at the root of the heap which improves its efficiency drastically. There is requirement of performing all the operations in logarithmic time.

The different data structures and their operations are given below.

Data Structures Used:

1. Minimum Heap: This data structure is used to keep track of the construction of buildings based on their execution time. The heap is constructed based on the execution times.
   The heap supports the following operations:
   i) insert (long totalTime, long totalExecutionTime, Red_Black_Node node) – inserts a heap node into the heap. It also contains a red black node as a reference to the red black tree node. The time complexities of the operations are as given below in table 1.
   Heap Node: contains the total time to be executed, total time a building has been constructed and the red black tree to which it points to.
   ii) reArrangeHeap () – Rearranges the heap after the insertion of heap nodes. This is done in *log(n)* time including the insert.
   iii) extractMin () – We extract the minimum from the heap and then restructure the heap which contributes to the amortized cost of *log(n)* for this operation.

   iv) remove (int position) – Removes the node present at the given position and restructures the heap accordingly. It also has the amortized cost of *log(n).*
   v) minHeapify (int givenIndex) – restructure the heap so that the minimum element is on top.

Operation of heap	Time Complexity
insert	O(log(n))
reArrangeHeap	O(log(n))

extractMin	O(log(n))
Remove	O(log(n))
minHeapify	O(log(n))

Table 1 Time complexities of the operations of minimum heap

2. Red Black Tree: This data structure is used as it gives efficient search capability. It has the red black nodes ordered by the building identifier. The red black tree has the following main operations:
   i) getBuildingDetails (int buildingID1, int buildingID2) – gets for all the buildings in the given range specified and gives a list of the buildings within the range. This operation is done in *O(log(n) + S)* as we must print all the triplets present in the given range.
   ii) getBuildingDetails (int buildingID) – gets for the building given and returns the building number if it is present in the tree or else returns 0. This involves a search where we might end up going up to the leaf and hence runs in *log(n)* time.
   iii) insert (int buildingID, long totalExecutionTime, long executedTime) – This inserts a building with all the details into the red black tree. This involves searching where the node must be inserted into and making node changes which is done in *log(n)* time.
   iv) delete (Red_Black_Node z) – deletes a node from the red black tree and rearranges the tree accordingly to satisfy the red black tree condition.

Operations of Red black tree	Time Complexity
Search in range	O(log(n))
search	O(log(n))
Insert	O(log(n))
delete	O(log(n))

Table 2 Time Complexities of the operations of the red black tree

Assumptions:

1. A building is taken for building only after five days or after its execution time is complete whichever is least.
2. Once a building is complete, print the building and the global day counter at which it has completed.

3. If there is a print building command at the same time a building is being deleted (due to its completion) then we need to print all the buildings requested by the print building command and then print the deleted building.
4. If a print building command is given for a building which has not been inserted then a line "(0,0,0)" is inserted at that point in the output file.
5. If there is print building which asks for a range of buildings and there no buildings present within the range, then it prints a line as shown above.
6. Any number of buildings can be inserted into the city.
7. If a building is in its last iteration where the executed time is not a multiple of 5, then we immediately start constructing the next building got from the heap.
8. If there is an insert of a building with same building identifier then a line "Duplicate building number being insert, Aborted construction" is written to the file a RuntimeException () is thrown and the construction is stopped.

There are different other supporting classes used like:

1) *Building:* This contains all the operations that can be performed on buildings and has the references to the heap and the red black tree.
2) *Heap Node and Red Black Node:* Serves as the nodes which contain building information for the heap and red black tree respectively.
3) *risingCity:* This contains the core logic of when which buildings must be constructed and input output operations. A global day counter runs in here which keeps track of all the different operations. The flow chart of the logic is given below in Figure 1:

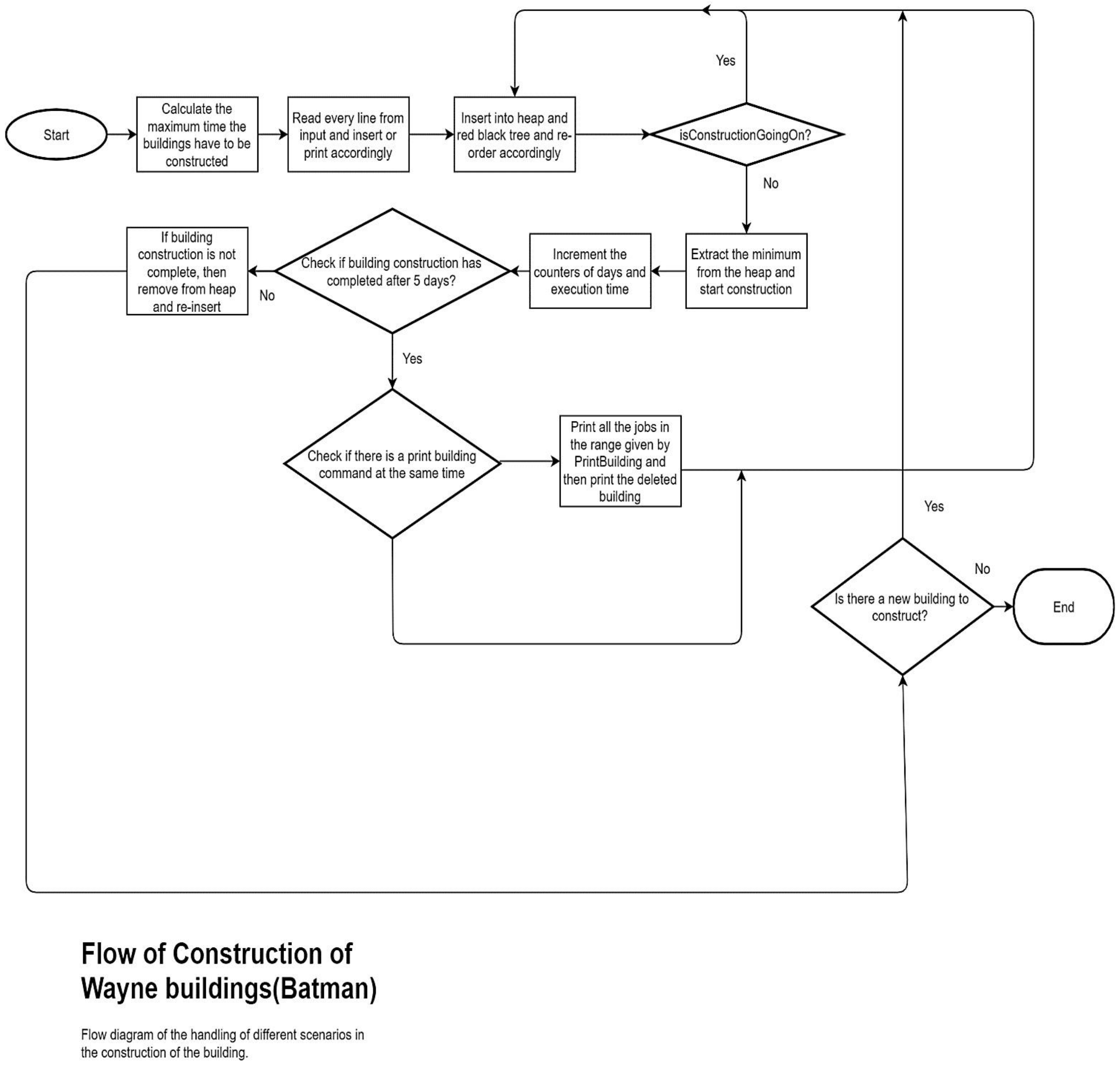
Start
Calculate the maximum time the buildings have to be constructed
Read every line from input and insert or print accordingly
Insert into heap and red black tree and re-order accordingly
isConstructionGoingOn?
Yes
No
Extract the minimum from the heap and start construction
Increment the counters of days and execution time
Check if building construction has completed after 5 days?
No
If building construction is not complete, then remove from heap and re-insert
Yes
Check if there is a print building command at the same time
Print all the jobs in the range given by PrintBuilding and then print the deleted building
Yes
Is there a new building to construct?
No
End
Flow of Construction of Wayne buildings(Batman)
Flow diagram of the handling of different scenarios in the construction of the building.

Figure 1 Flow chart of the sequence of operations in the main function

Conclusions:

The combined use of minimum heaps and red black trees gives a great performance advantage for the construction of buildings where the building is to be constructed (lowest executed time) is extracted from the heap which was designed specifically for this and the search is done on the Red black tree which is known for the efficiency of its search.