Priority Queue(Max Heap):-

I have implemented it using Arraylist .

Extra class: Pair :- I have used this for tackeling FIFO order.

Insertion:-

Here we first simply add the element to the end and then start heapifying it I.e.

First get its parent and then check whether the the child is bigger as compared to parent or not if bigger then Swap both of them and else break.

ExtractMax:-

Normally:-

I will remove the first element in array and set first element which is removed equal to last element in array and delete the last element.

Then call a function delete which is heapifying.

Delete:-

I have maintained four indexes namely max,parent(p),left child(lc),right child(rc). I assign them initially as 0,0,1,2 now I iterate , and let initially set max =lc and then check that is right child priority is higher than left childs, if so then max=rc.

Here one special case is of same priority :-

Here we need to maintain the FIFO order if priority is same and I am maintained it by having Pair as my node as In pair I store the serial number in which elements have been inserted.

So for same priotiy we check for the serial number also and assign max to that which comes earlier.

Then after getting max then I check it with its parent and if parent' s priority is higher then I swap both of them and reassign them as p=max,lc(2*p+1),rc=(2*p+2) and here also the same priority case can arrive which is handelled in the same way as above by checkin the serial number and if p has come letter then swap them and reassign lc and rc as above. Else break.

NOTE:- I have overloaded the insert method and also have a extractmax also ,This is for use in ProjectManagement(i.e. FIFO in projectmanagement).

Trie :-

TrieNode:-

In this we maintain a list of childrens of a node and Boolean isEndofWord , to mark a end of the word.

Basic:-

We start adding everything after the root i.e. there is nothing at root it's just a starting node.

Here I am maintaining a level for every node.

Insert:-

It takes two argument word and object .

Here I start at root and move down the trie .Iteration is over the word length and index is calculated by using the characters .Now we check whether there in the array of our node that index's node exists or not ,if it does not exist i.e. it is null then create a new Node there and if not then go to its children's array and increase the level.

Now if we got out of the for loop and nodes isEndofword is true that means there is no insertion and it returns false there.But if (the loop index i) equals the word length that means we have travelled all the way to that node here I store my nodes value and mark Endofword=true.

Search:-

It simply does the iterative searching.We start from root and search.If in iteration there is no node is find at a index(calculated from the word's charcters) then we return null i.e. it does not have that word else we search in its childrens array.

Now after getting out of iteration if your node remains null then the word is not there else return the node.

Delete:-

Edge case if word is null or root is null then return false .

Then there is a recursive function delrec(String,node,level)

Delrec:-

Base case :- if level equals the word's length. In this if your node you want to delete have child then you just unmark the isEndofword for that node else we just set that node to null.

Recursion:-

Now I get the index from my word and using level . If the node is not null then I hae a new node which I set to that node's child.now if this node is null then deletion is not happened and if not then delete on level +1 with node as this new node as above . now if this recursion gives true then you set your node at that index as null and similar case is checked as above.

Startswith:- This is similar to the searching .

Printing functions:-We have used different traversal like preorder.

REDBLACKTREE:-

Insert:- Simple BST insert then a recursive function fix for fixing the violation of color and height.

(Cases are handelled in code)

Search:- Simple binary tree searching.

PROJECTMANAGEMENT:-

Data Structures used:-MaxHeap,RBTree,ArrayList,Trie.

Classes :-

Schedule_Driver:-All the functions implemented as recquired.

Job ,project, user(ToString method are overridden as per recquirement for printing purpose.)

Implementation :- I am maintaining two arraylist for finished and notfinished jobs.

Time Complexity:-

MaxHeap:-

N is number of nodes.

Insert:- O(logn)

ExtractMax:- O(logn) Deletion recquires (if less number of nodes with same priority) O(1) time for deletion but now reordering them recquires O(logn) steps

Trie:- M is no. of TrieNode's stored in the trie

Insert & search:- O(N) (N is key length)

Delete:-O(key length)

PrintLevel:- O(M),the recursive function "pl" in printlevel do preorder traversal of trienode and steps recquired are always less than or equal to M.

PrintTrie:-O(M)number of steps recquired are (level*no.of words)which is less than or equal to M.

Print:-(M*levels):-As print calls Printlevel for each Level.

RedBlackTree:-

n is number of nodes

Insert:-(O(logn)) Insertion and Fixing both take O(h) steps (h=height which is O(logn)).

Search:-O(logn)