OBJECTIVE : To create a job scheduler using Red Black and Min heap.

TOOLS : Programming language – C++

MIN HEAP : Job entering the min heap is heapified on the basis of lowest executed_time of a job so far. When a job enters the processor it is executed for 5 ms or the remaining that is needed for its completion.

FUNCTIONS :

- void heapify() : It rearranges heap to maintain heap property.
- void inserth(T ,T) : It is used to insert values in Min heap. The jobid and total time is passed in the parameters as template object.
- void swap(long child, long parent) : This function is used to swap child with parent based on its values. It is called in Heapify.
- long getSize() : It is used to calculate the current size of heap and returns the value.
- void setSize(long) : It doubles the size of the heap.
- Remove() : It removes the minimum value in the min heap.
- void afterdel() : It rearranges the heap when an element is deleted.
- void display() : It prints the values currently in the min heap.

RED BLACK TREE : This tree is arranged with respect to job id of the jobs.

FUNCTIONS

- void rbinsert(NODE *treeroot, long jid, long ttime) : It inserts values of job id and total time in the red black tree.

- void rbinsertfix(NODE *treeroot, NODE z) : It is used for rebalancing red black tree after a new insert.

- void rrotate(NODE *treeroot, NODE n) : It is used for the right rotation of the tree.

- void lrotate(NODE *treeroot, NODE n) : It is used for the left rotation of the tree.

- void rbprintjob(NODE root, long k, int flag) : This function checks that for a value ‘k’ any job id exists or not. If yes, that job id along with it executed time and total time is written in output file; else flag is set to 0 and (0,0,0) is written in file.

- NODE maxn(NODE root) : It returns the node with maximum job id in the tree.

- NODE minimum(NODE root) : It returns the node with minimum job id in the tree.

- void rbprevjob(NODE*root,long k) : It calculates the largest value smaller than k.

- void rbnextjob(NODE *root, long k) : It calculates the smallest value larger than k.

- void rbupdateextime(NODE root, long jid, long time) : It updates the executed time of a job in red black tree.

- long rbprintjob1(NODE root,long flag,long val1,long val2) : It calculated the values of job id in the range val1 and val2 and writes them in the file or else flag is set to 0 and (0,0,0) is written in the file.

- void porder(NODE n,long min,long max,long r1,long r2) : This function is called due to rbprintjob1(). It checks whether the values of jobid lies in that range or not. It stores the jobid lying in that range in a vector. This function implements BST.

- NODE search(NODE root, long k) : It searches whether a jobid with value ‘k’ exists or not.

- void inorder(NODE n) : It displays values in red black tree in inorder form.

Structure:

Firstly the input file is read in a vector.

Then a function calculate() is called. The calculate() runs until the vector is empty. In each iteration the value of arrival time, command, jobid and total time is calculated.

A global variable progcounter is initialized with value 0 and is increased by 1 with each iteration. Progcounter keeps track of how many lines have been executed.

If the value of the arrival time in the file matches with the progcounter then according to that command functions of Insert, PrintJob , NextJob and PrevJob are called respectively. In insert command heap.inserth() is called and values are inserted in Min heap. Similarly rbinsert() is called to insert values in the Red Black tree. If PrintJob(val1 ,val2) is called then we first calculate the maximum job id and minimun jobid using functions NODE maxn () and NODE minimum() ; and check if val2 is less than the minimum jobid or val2 is greater than the maximum jobid we return null or else traverse the tree in O(log n+S) with help of binary search. Similar search is done in case of PrintJob(val) and is found in O(log n) time. If there is no arrival time for a particular progcounter then execution time of the job is increased and the execution value is also updated in Red Black tree by rbupdateextime(). A particular job is booked for 5ms or its remaining time. If a new value is inserted inbetween that time then the newly inserted node is stored at the last position of the heap and when that job is completed; heapify() is called and the similarly the value is updated in Red black tree using function rbupdateextime().