CS 2810
ADVANCED PROGRAMMING LAB

IMPLEMENTATION OF LEFTIST HEAPS

TEAM:

Madhavi Y	*CS09B025*
D Vamsi Krishna	*CS09B006*
M Uday Chowhan	*CS09B006*

Files included:

1. leftist_tree.cpp
2. leftist_heap.h
3. rand.cpp

- leftist_heap.h contains declarations of the functions used in of various functions used in the heap class.
- leftist_tree.cpp has the definitions and the main function.
- rand.cpp outputs random values into input files

Description:

- The leftist heap is implemented using lists
- Node of the list is lhnode
- Each node has data , rchild , lchild

Functions in the class

- return_root() : This function returns the root of the heap
- insert : This function inserts an element into the heap
- extract_min : This function extracts the minimum from the heap
- sort : This function sorts the heap

Functions outside the heap class

- swap : Function which swaps two nodes
- meld : Function which melds two heaps
- inorder: Function which prints the element of a heap in inorder traversal

Pseudocode for functions in the leftist_heap:

Pseudocode for meld operation

// to meld two heaps

```
meld (rA,rB)
    if  rA = NULL  return rB
    if  rB = NULL  return rB
```

```
      if  data(A) < data(B)
            swap A,B
      right (A)  <=  meld(right A, rB)
      if  dist (right A)  >  dist (left A)
            swap  right A and left A
      if  right A = NULL
            dist A = 0
      else
            dist A = 1+ dist right A
      return A
```

Psedocode for insert operation

// To insert an element into the heap

```
insert (num , A)
      Node .data = num
      meld ( rA,Node)
```

Psedocode for extract_min operation

//To extract an element from the heap

```
extarct_min( A )
      meld ( right A, left A)
```

Pseudocode for sort

// To sort the heap given a min heap

```
sort ( array)
      while root!=NULL
            push extarct_min into array
```

Pseudocode for inorder traversal

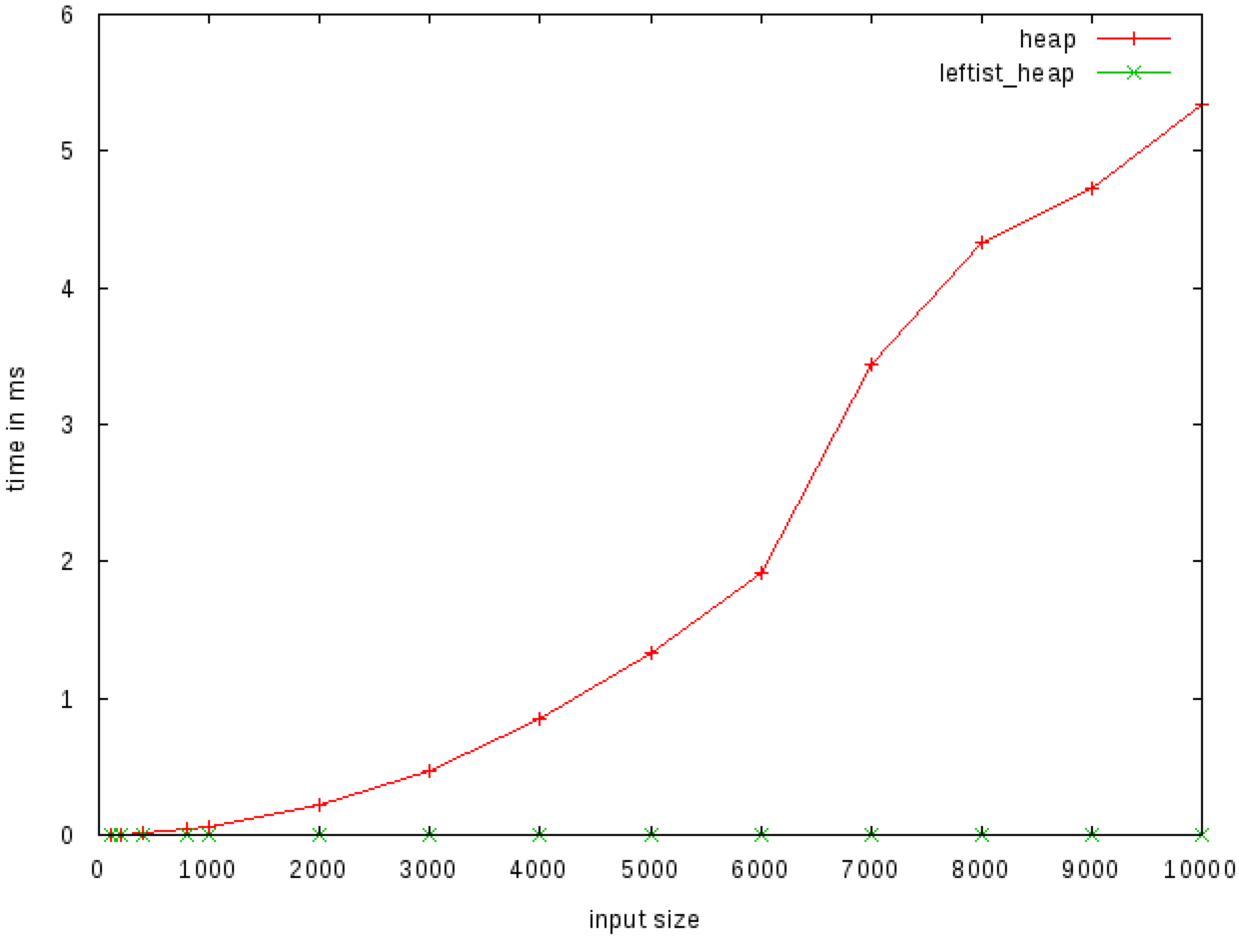
//To print the elements in inorder into a file

```
inorder ( root )
      if left_root !=NULL
            inorder (left_root)
      print root.data
      if right_root!= NULL
            inorder(right_root)
```

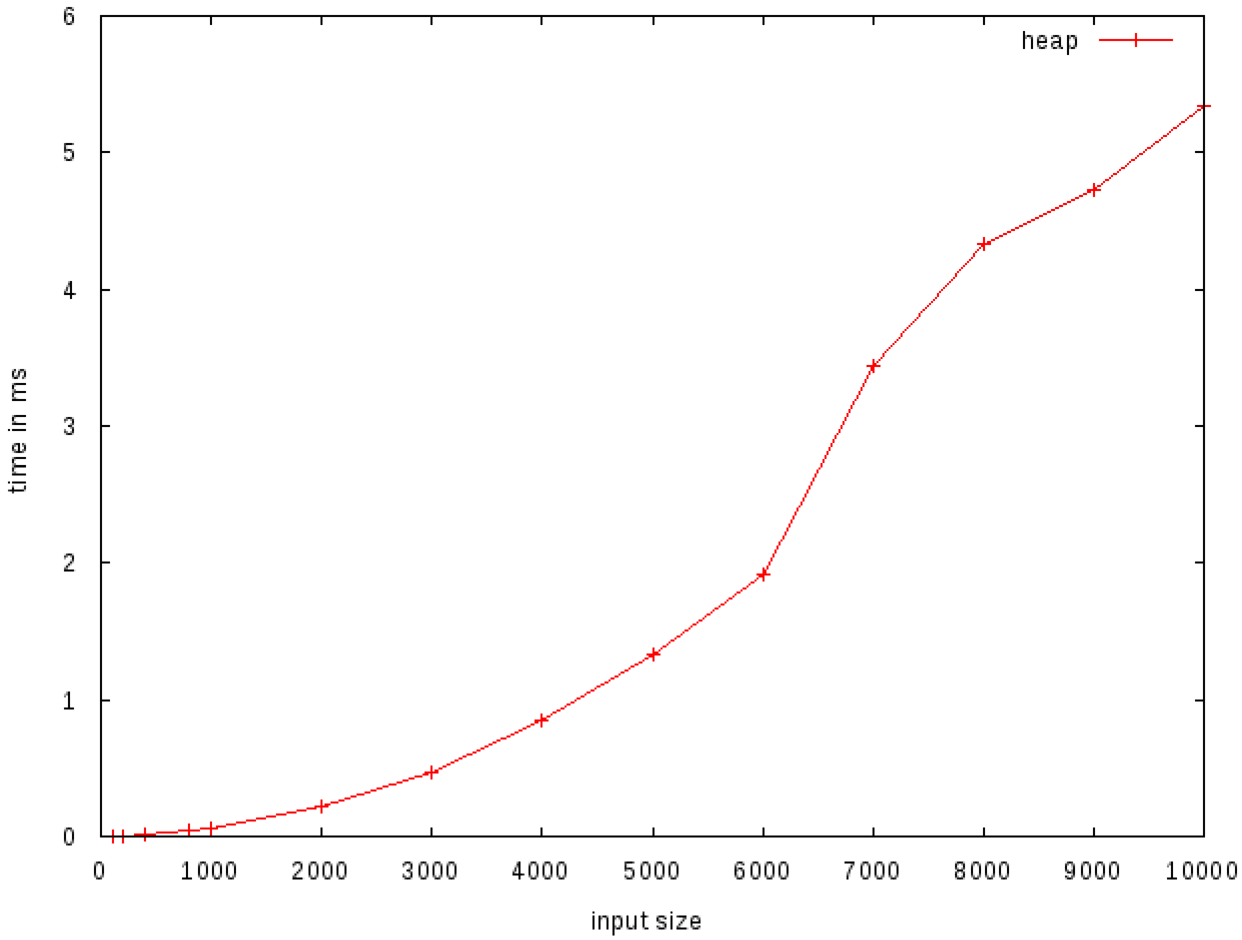
Time taken for meld operation:

size	*time in ms for heaps*	*time in ms for leftist heaps*
100	0.001296	0.0000002
200	0.004242	0.0000003
400	0.009356	0.0000003
800	0.038538	0.0000003
1000	0.057538	0.0000003
2000	0.213878	0.0000003
3000	0.474614	0.0000004
4000	0.851594	0.0000004
5000	1.32907	0.0000004
6000	1.91452	0.0000004
7000	3.43931	0.0000004
8000	4.33397	0.0000004
9000	4.72265	0.0000004
10000	5.34456	0.0000004

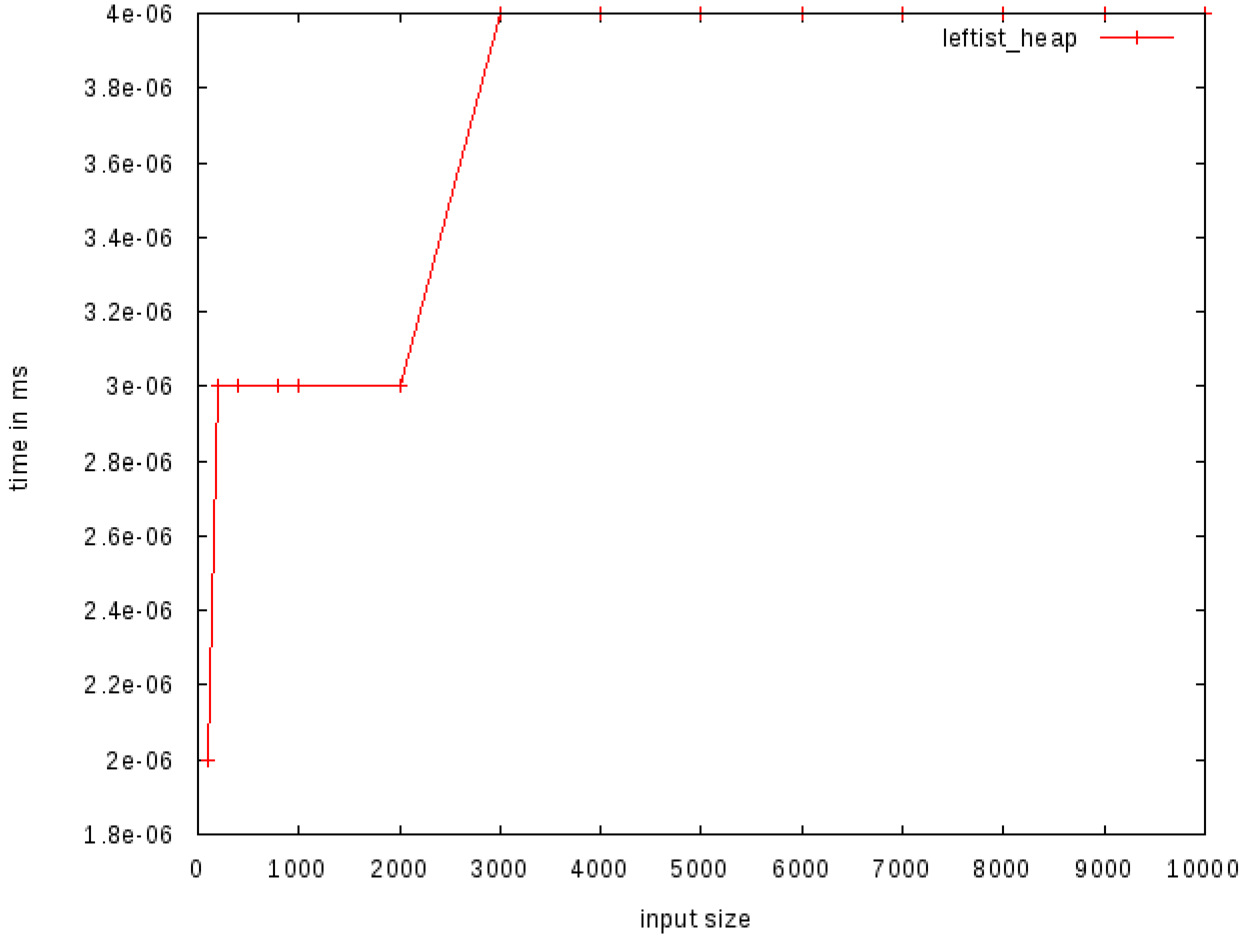
Input size vs time graph comparision graph:



Inputsize vs time (using heaps):

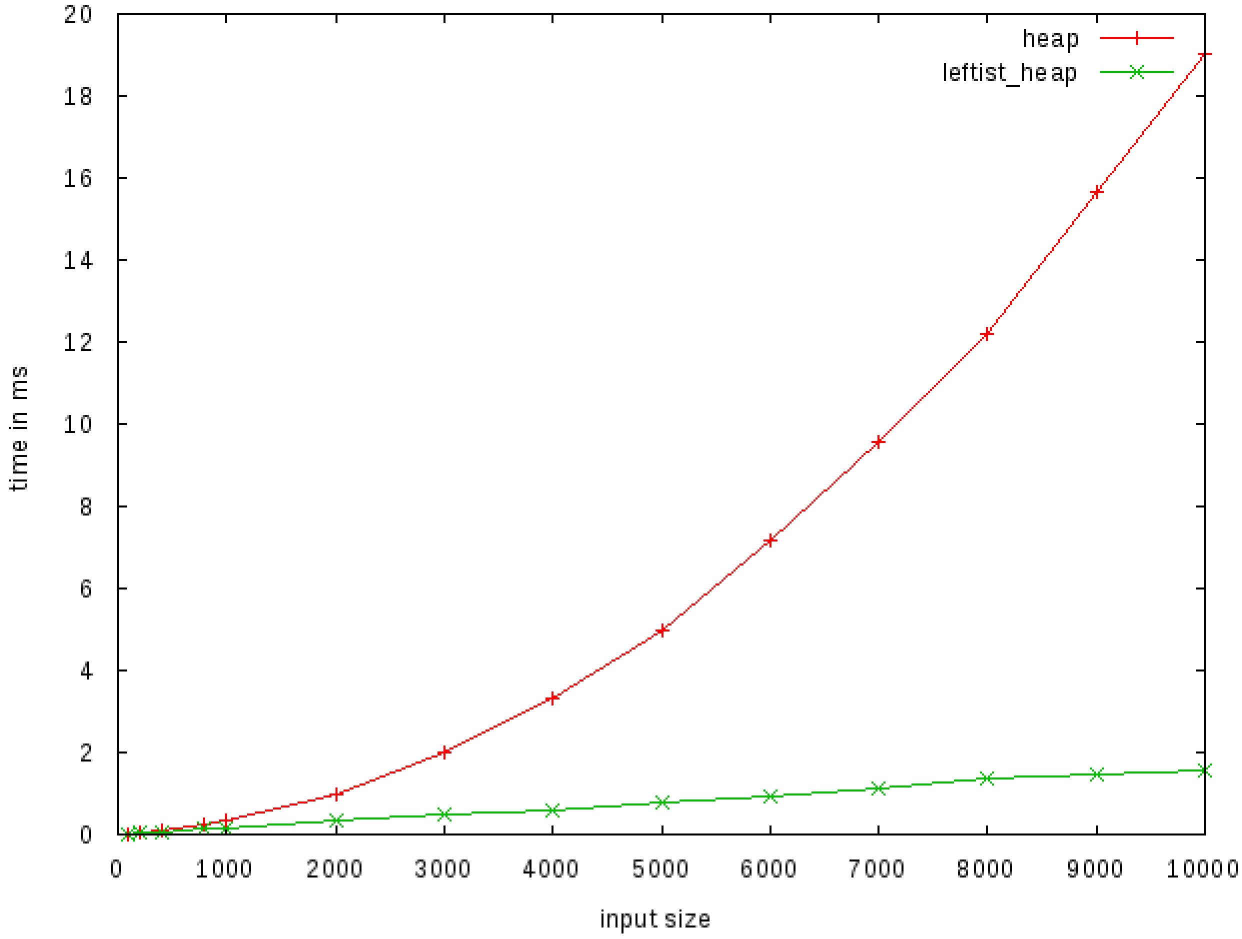


Inputsize vs time using (leftist heaps):



Time taken for insert operation:

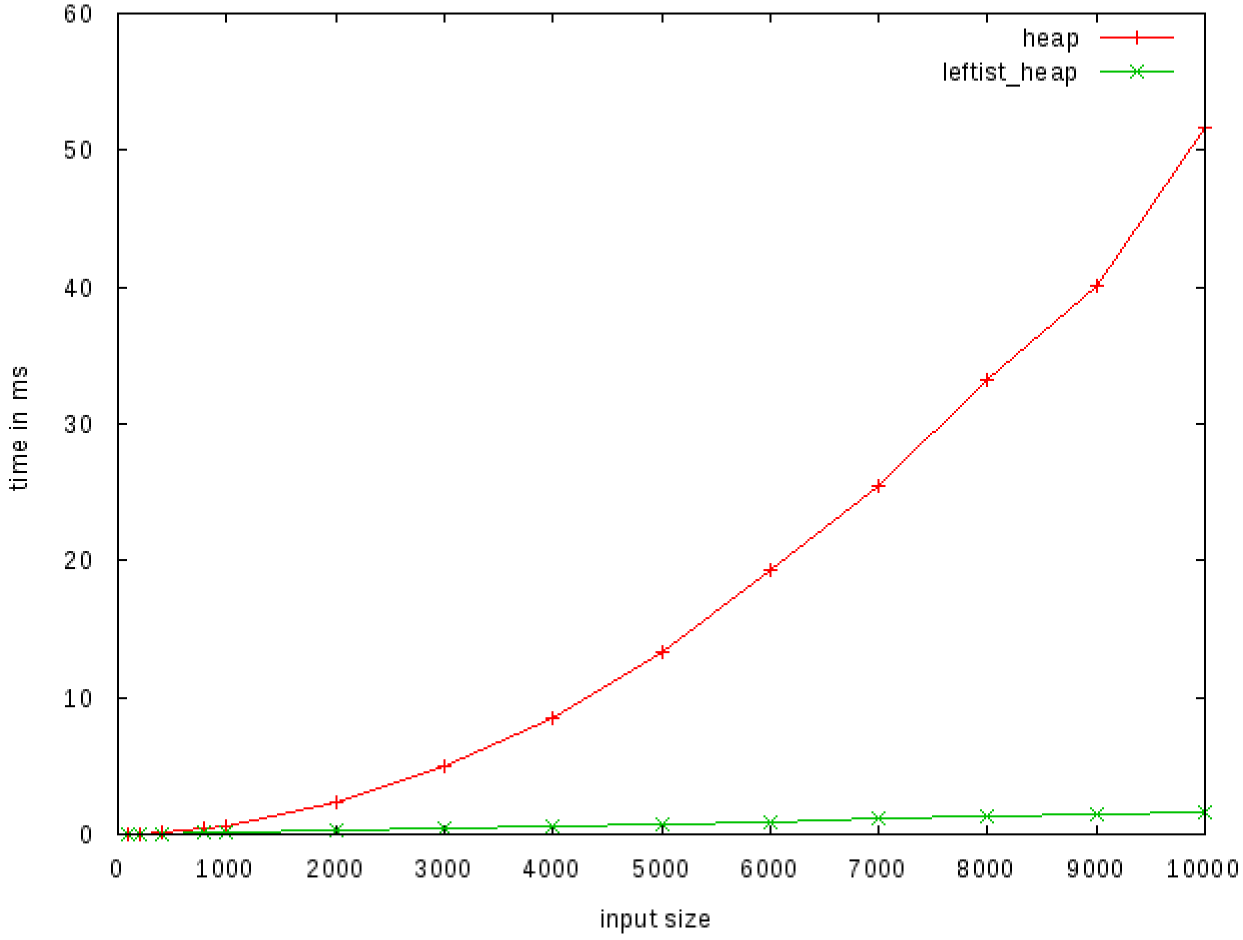
size	*time in ms for heaps*	*time in ms for leftist heaps*
100	0.022146	0.021997
200	0.049293	0.042659
400	0.107539	0.06762
800	0.247986	0.140028
1000	0.340423	0.16192
2000	0.99037	0.326395
3000	1.97566	0.480265
4000	3.31273	0.59347
5000	4.99009	0.774384
6000	7.19431	0.931363
7000	9.54533	1.10795
8000	12.1873	1.34162
9000	15.6662	1.48778
10000	19.0074	1.57642



Time taken for extract_min operation:

size	*time in ms for heaps*	*time in ms for leftist heaps*
100	0.015507	0.016789
200	0.053951	0.043672
400	0.141041	0.065796
800	0.433042	0.125728
1000	0.643976	0.151981
2000	2.29451	0.319467
3000	4.91096	0.48750

4000	8.43273	0.48750
5000	13.266	0.781987
6000	19.2514	0.964995
7000	25.4681	1.12189
8000	33.2204	1.31197
9000	40.1456	1.46786
10000	51.6365	1.58207


heap
leftist_heap
time in ms
input size