COMP 2611 – Data Structures

Lab #7

Heaps

Download the `Lab7-Files.zip` file from myeLearning. The zipped file contains a Dev-C++ project, `MaxHeap.dev`. The project contains the code for all the functions from Lab #6, Part 2 as well as the code for the following functions discussed in Lectures 13-14:

Function	**Description**
void buildMaxHeap(MaxHeap * heap);	Converts *heap* to a max-heap.
int maximum (MaxHeap * heap);	Returns the biggest value in the max-heap.
int deleteMaxHeap (MaxHeap * heap, int i);	Deletes the node at index *i* from the max-heap. The value stored at index *i* is returned.
void insertMaxHeap (MaxHeap * heap, int data);	Inserts a new value, *data*, in the max-heap.
void deleteAllMaxHeap (MaxHeap * heap);	Deletes all the nodes from a max-heap.
void heapSort (int A[], int numElements);	Sorts array *A* using the heap sort algorithm.

(1) Write the code for the *maxHeapify* function in `MaxHeap.cpp`. The prototype for *maxHeapify* is as follows:

```
void maxHeapify (MaxHeap * heap, int i);
```

Use the algorithm that was given in Lecture 13:

```
maxHeapify (MaxHeap * heap, int i) {
   left = i * 2;
   right = i * 2 + 1;
   largest = index of largest of:
                  heap->A[i],
                  heap->A[left],
                  heap->A[right]

   if (largest != i) {
      swap heap->A[largest] with heap->A[i];
      maxHeapify(heap, largest);
   }
}
```

When finding the largest of the values stored at location *i*, *left*, and *right*, remember to deal with the case where Node *i* does not have a left child or a right child.

(2) The field, *A*, of a 'max-heap' contains the following values:

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
	99	47	84	68	76	80	51	66	52	70	71	57

(a) Draw the nodes of the 'max-heap'.

(b) All the nodes of the 'max-heap' except two satisfy the max-heap property. Find the nodes that do not satisfy the max-heap property.

(c) Draw the max-heap after the *maxHeapify* algorithm has executed on the parent of the nodes from (b).

(d) The file, `MaxHeap-Q2.txt`, contains the values to be stored in the array *A*. In `UsingMaxHeap.cpp`, create the 'max-heap' from the values in the file (use the function, *initMaxHeapFromFile*).

(e) In `UsingMaxHeap.cpp`, call the *maxHeapify* function with the parent of the nodes from (b). Display the values in the max-heap after the call to *maxHeapify*. Verify that it is the same max-heap that you obtained in (c).

(3) The field, *A*, of a max-heap contains the following values:

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
	85	81	72	63	75	70	65	50	60	74	40

(a) Draw the nodes of the max-heap.

(b) Draw the max-heap after Node 2 is deleted.

(c) In `UsingMaxHeap.cpp`, create the max-heap using the function, *initMaxHeapFromArray*.

(d) In `UsingMaxHeap.cpp`, call the *deleteMaxHeap* function with Node 2. Display the values in the max-heap after the call to *deleteMaxHeap*. Verify that it is the same max-heap you obtained in (b).

(4) The field, *A*, of a max-heap contains the following values:

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
	50	48	40	41	33	22	38	35	28	30	25	15

(a) Draw the nodes of the max-heap.

(b) Draw the max-heap after inserting the value 75.

(c) In `UsingMaxHeap.cpp`, create the max-heap using the function, *initMaxHeapFromArray*.

(d) In `UsingMaxHeap.cpp`, call the *insertMaxHeap* function with the value 75. Display the values in the max-heap after the call to *insertMaxHeap*. Verify that it is the same max-heap you obtained in (b).

(5) An integer array, *A*, contains the following values:

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
60	20	41	15	25	75	30	35	50	23	38	45

In `UsingMaxHeap.cpp`, sort the values in *A* using the function, *heapSort*. Display the values in the array after the function is called to verify that the values have been sorted.

NB: The values in array *A* start from location 0 but the values in the max-heap start from location 1.

End of Lab #7