CptS 223 – Advanced Data Structures in C++

Written Homework Assignment 6: Heaps and Sorting

Assigned: Wednesday, December 2, 2020
Due: Friday, December 11, 2020

I. Problem Set:

1. **(8 pts)** Time for some heaping fun! What's the time complexity for these functions in a C++ STL binary heap of size N?

Function	Big-O complexity
push(x)	O(logn)
top()	O(1)
pop()	O(logn)
buildHeap(vector<int>{1...N})	O(n)

2. **(4 pts)** What is a good application for a priority queue (a binary heap)? Describe it in at least a paragraph of why it's a good choice for your example situation.

Priority queues can be used in Dijkstra's Algorithm to find the shortest path. This can be useful for planning out routes in a GPS and which flights to take to get from one state to the other. If someone wanted to get from place A to place B, using priority queues can help the GPS or flight schedule plan out the shortest path to get there.

3. **(5 pts)** For an entry in our binary heap (root @ index 1) located at position i, where are it's parent and children?

   Parent: i/2

   Children: Left child(i) = 2i and Right child(i) = 2i + 1

4. **(9 pts)** Show the result of inserting 10, 12, 1, 14, 6, 5, 15, 3, and 11, one at a time, into an initially empty binary heap. Use a 1-based array like the book does.

After insert(10):

10										

After insert (12):

10	12									

etc:

1	12	10								

1	12	10	14							

1	6	10	14	12						

1	6	5	14	12	10					

1	6	5	12	10	14	15				

1	3	5	6	12	10	15	14			

1	3	5	6	12	10	15	14	11		

5. **(3 pts)** Show the same result (only the final result) of calling buildHeap() on the same vector of values: {10, 12, 1, 14, 6, 5, 15, 3, 11}

1	3	5	6	12	10	15	14	11		

6. **(6 pts)** Now show the result of three successive deleteMin / pop operations from the prior heap:

3	6	5	11	12	10	15	14			

5	6	10	11	12	14	15				

6	11	10	15	12	14					

7. **(5 pts)** What are the average complexities and the stability of these sorting algorithms:

Algorithm	Average complexity	Stable (yes/no)?
Bubble Sort	O(n^2)	Yes
Insertion Sort	O(n^2)	Yes
Merge Sort	O(nlogn)	Yes
Radix sort	O(nd) (d is the number of digits in the largest number)	Yes
Quicksort	O(nlogn)	No

8. **(3 pts)** What are the key differences between Mergesort and Quicksort?

The key differences between Mergesort and Quicksort is that MergeSort is a stable sorting algorithm with Quicksort is not. In addition, MergeSort divides the array into two sub-arrays (n/2) while Quicksort divides the array into two sub-arrays around the "pivot". Mergesort also is more efficient and works faster than Quicksort in larger array sizes, while Quicksort is more efficient and works faster in smaller array sizes.

9. (6 pts) Draw out how Mergesort would sort this list:

24	16	9	10	8	7	20

24 16 9 10 8 7 20
24 16 9 10 8 7 20
24 16 9 10 8 7 20
16 24 9 10 7 8 20
9 10 16 24 7 8 20
7 8 9 10 16 20 24

10. (6 pts) Draw how Quicksort would sort this list:

24	16	9	10	8	7	20

Let me know what your pivot picking algorithm is (if it's not obvious): 10

24 16 9 10 8 7 20
24 16 9 20 8 7 10
7 16 9 20 8 24 10
7 8 9 20 16 24 10
7 8 9 16 20 24 10
7 8 9 10 16 20 24

II. Submitting Written Homework Assignments:

1. On your local file system, create a new directory called HW6. Create/Move your HW6.pdf file in the directory. In your local Git repo, create a new branch called HW6. Add your HW6 directory to the branch. Add and commit HW6.pdf. Merge HW6 branch with the Master Branch and push the Master branch to the remote origin.

III. Grading Guidelines:

This assignment is worth 55 points. We will grade according to the following criteria:

- See above problems for individual point totals.