Lab 3

Quicksort

Quicksort, like merge sort, applies the divide-and-conquer paradigm. It is a three step divide-and-conquer process for sorting a typical subarray A[p...r]:

Divide: Partition the array A[p...r] into two subarrays A[p...q-1] and A[q+1...r] such that each element of A[p...q-1] is less than or equal to A[q], which is in turn less than or equal to each element of A[q+1...r]

Take a look at the PARTITION procedure.

PARTITION(*A*, *p*, *r*)

1. $x \leftarrow A[r]$
2. $i \leftarrow p - 1$
3. **for** $j \leftarrow p$ **to** $r - 1$
4. **do if** $A[j] \leq x$
5. **then** $i \leftarrow i + 1$
6. exchange $A[i] \leftrightarrow A[j]$
7. exchange $A[i + 1] \leftrightarrow A[r]$
8. **return** $i + 1$

Conquer: Sort the two subarrays A[p…q-1] and A[q+1…r] by recursive calls to quicksort.

Take a look at the following procedure.

QUICKSORT(*A*, *p*, *r*)

1. **if** $p < r$
2. **then** $q \leftarrow$ PARTITION(*A*, *p*, *r*)
3. QUICKSORT(*A*, *p*, *q* - 1)
4. QUICKSORT(*A*, *q* + 1, *r*)

Combine: The two subarrays are already sorted. So no work is needed to combine them.

HEAPSORT

Heapsort is a comparison-based sorting algorithm to create a sorted array (or list), and is part of the selection sort family.

Steps involved:

1. Build a heap from the data
2. Remove the largest element from the heap and place it in the array. Reconstruct the heap and remove the next element to insert it in the array. Repeat until all the elements are removed from the heap.

Pseudo Code:

HEAPSORT(*A*)

1. BUILD-MAX-HEAP(*A*)
2. **for** *i* ← *length*[*A*] **downto** 2
3. **do** exchange *A*[1] ↔ *A*[*i*]
4. *heap-size*[*A*] ← *heap-size*[*A*] - 1
5. MAX-HEAPIFY(*A*, 1)

BUILD_MAX_HEAP procedure:

BUILD-MAX-HEAP(*A*)

1. *heap-size*[*A*] ← *length*[*A*]
2. **for** *i* ← ⌊*length*[*A*]/2⌋ **downto** 1
3. **do** MAX-HEAPIFY(*A*, *i*)

MAX_HEAPIFY procedure:

MAX-HEAPIFY(A, i)

1. l ← LEFT(i)
2. r ← RIGHT(i)
3. if l ≤ heap-size[A] and A[l] > A[i]
4. then largest ← l
5. else largest ← i
6. if r ≤ heap-size[A] and A[r] > A[largest]
7. then largest ← r
8. if largest ≠ i
9. then exchange A[i] ↔ A[largest]
10. MAX-HEAPIFY(A, largest)