**50 points. 75 minutes. Closed book/notes.** **Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. For each of the following program fragments, determine the time complexity using the notation. Explain your analysis next to each program fragment

|  |  |  |  |
| --- | --- | --- | --- |
|  | Program Fragment | Time Complexity  Notation) | Explanation |
| 1 | for (int i=0; i<2\*n; i++){  for (int j=0; j<n/2; j++){  a = a + i + j;  }  } |  |  |
| 2 | for (int i=0; i<n; i++){  a = a + i;  }  for (int j=0; j<n; j++){  a = a + j;  } |  |  |
| 3 | for (int i=0; i<n\*\*3; i++){  for (int j=0; j<2\*n; j++){  a = a + i + j;  }  } |  |  |
| 4 | for (int i=0; i<n; i++){  for (int j=0; j<i; j++){  a = a + i\*j;  }  } |  |  |
| 5 | for (int i = 1; i < n; i \*= 2)  {  a = a + i;  } |  |  |

1. Show the operation of sorting the following sequence using mergesort.

**Arr = [ 4, 14, 25, 53, 96, 29, 19]**

The mergesort algorithm is given below.

MERGE-SORT (A, temp, p , r)

if p < r

q = |\_ (p + r) / 2 \_|

MERGE-SORT (A, temp, p , q)

MERGE-SORT (A, temp, q + 1, r)

MERGE (A, temp, p, q, r)

//////////////////////////////////////////////

MERGE (A, temp, p, q, r)

// merge A[p..q] with A[q+1..r]

i = p

j = q + 1

// copy A[p..r] to temp[p..r]

for k = p to r

temp[k] = A[k]

//merge back to A[p..r]

for k = p to r

if i > q // left half empty, copy from the right

A[k] = temp[j]

j = j + 1

else if j > r // right half empty, copy from the left

A[k] = temp[i]

i = i + 1

else if temp[j] < temp[i] // copy from the right

A[k] = temp[j]

j = j + 1

else

A[k] = temp[i] // copy from the left

i = i + 1

1. Given the following max-heap, show the operation of sorting using heapsort.

The heapsort algorithm is given below.

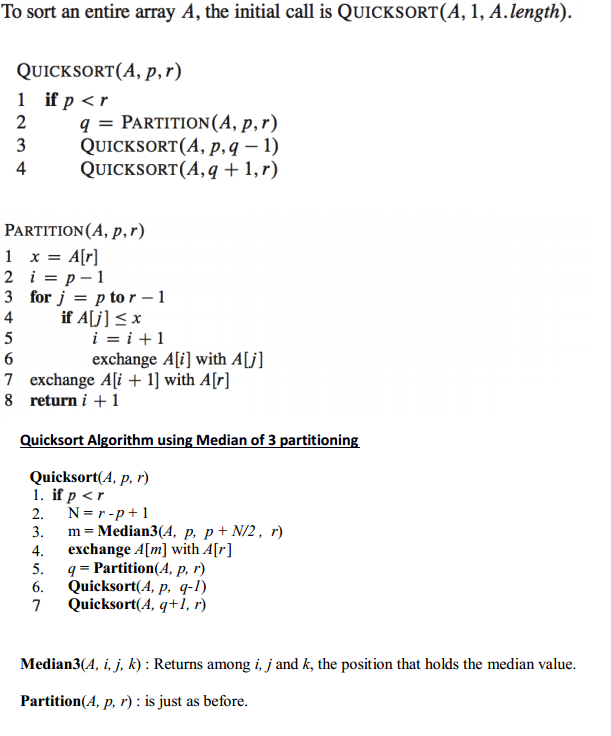
|  |  |
| --- | --- |
|  |  |
|  |  |

|  |
| --- |
|  |
|  |

1. Show the operation of sorting the following sequence using quicksort.

**Arr = [ 27 ,11, 10, 188, 335, 822, 199, 379, 171 ]**

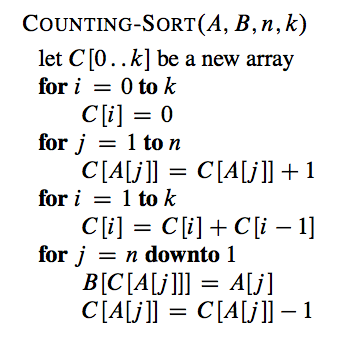
The quicksort algorithm is given below.



1. Illustrate the operation of counting sort on the following sequence;

**Arr =[5, 0, 2, 0, 1, 3, 4, 5, 1, 3, 2, 5]**

The algorithm for counting sort is given below.



1. How many different methods we can use to choose the ***pivot*** value for the quicksort algorithm? Explain them
2. Explain the relationships (similarities/differences) between *tree, binary tree* and *heap* data structures