HOMEWORK-4
Total Points: 70

1. [15 points] What would be the time complexity of the quick sort implementation discussed in class for the case when all the elements of the array have the same value? Justify your answer.

   If we run the quick sort algorithm discussed in the class when all the inputs are the same, the partition algorithm will roughly return the middle index all the time. Therefore, the recurrence relation for the time complexity is:

   $T(n) = 2T(n/2) + c_1 * n + c_2$
   $T(1) = c$

   Solving using expansion method:
   $T(n) = 2(2T(n/4) + c_1 * (n/2) + c_2) + c_1 * n + c_2$
   $= 2^2T(n/2^2) + 2c_1n + c_2(1+2)$
   $= 2^2(2T(n/2^3) + c_1 * (n/2^2) + c_2) + 2c_1n + c_2(1+2)$
   $= 2^3T(n/2^3) + 3c_1n + c_2 (1+2+2^2)$
   The kth term is
   $T(n) = 2^kT(n/2^k) + kc_1n + c_2 (1+2+2^2 + \ldots + 2^{k-1})$
   $= 2^kT(n/2^k) + kc_1n + c_2 (2^k - 1)$
   Substituting $n = 2^k$
   $T(n) = nT(1) + c_1 * \log_2 n * n + c_2(n-1) = O(n * \log(n))$

   Full marks if the justification and derivation of time complexity are correct. Otherwise, give zero.

2. [20 points] Modify the randomized quicksort algorithm discussed in class to find the median of n array elements without always sorting the entire array. You will lose marks if your algorithm does unnecessary computation. What is the worst-case time complexity of your algorithm?

```
int median(int arr[], int lo, int k, int hi)
{

  int idx = partition(arr, lo, hi);

  if (idx == k) {
    return arr[idx];
  }
```

```
  if (idx < k) {
    return median(arr, idx+1, k, hi);
  }
  else {
    return median(arr, lo, k, idx-1);
  }
}
```

Initially, median takes the array (arr), the starting index 0 (lo), the index of the median element in the sorted array n/2 (k), and the index of the last element n-1(hi).

The partition function is the same as in the quick sort algorithm. It returns an index i such that all the elements at indices less than i are smaller than or equal to the arr[i], and all the elements at indices greater than i are larger than or equal to arr[i].

Now, if the index returned by partition, i, is the index of the median element, then we are done. On the other hand, if the i < k, then the median element is in the right subarray, i.e., i+1 to hi; otherwise, the median element is in the left subarray, i.e., lo to i-1. So, therefore unlike quicksort, in this case, only one recursive call is required.

Give 12 marks if the recursive calls are conditional, and only one recursive call is made at a given time. Otherwise, give at most 2 points.

In the worst case, the median can always partition the array into two parts, one is of size one, and the other is of size n-1.
Therefore the recurrence relation for time complexity is:

$T(n) = T(n-1) + c * n$
$T(1) = c_1$

Give five marks if the above recurrence relation is correct. Otherwise, give zero. If the derivation below is also correct, give three marks for that as well.

Solving recurrence:
$T(n) = T(n-1) + c * n$
$= T(n-2) + c*(n-1) + c* n$
$= T(n-3) + c* (n-2) + c*(n-1) + c*n$
The kth term is:
$T(n) = T(n-k) + c*(n + (n-1) + (n-2) + \ldots + (n-k+1))$
Substituting, $n-k = 1$

$T(n) = T(1) + c*(n + (n-1) + (n-2) + \ldots + 2)$
$= T(1) + c*(n + (n-1) + (n-2) + \ldots + 2 + 1 - 1)$
$= T(1) + c * (n(n+1)/2 - 1)$
$= O(n^2)$

3. [5 points] Give an algorithm with the worst-case complexity of O(n * log(n)) to find the median of n elements.

   We can use merge sort to sort all elements in O(n * log(n)) and then return the element stored at the middle index.

   Give zero marks if somebody says quick sort instead of merge sort.

4. [10 points] How many times will the merge procedure be called to sort the following sequence of integer elements using merge sort?

   `28 43 72 79 23 70 55 39 68 1 41 40 5 25 95 4 42 54 79 55`

   What would be the contents of the array every time the merge procedure returns? If the merge procedure is called k times, you need to write the contents of the whole array k times.

   The merge-sort is called 19 times.
   0.5 points for the correct values of the array after a merge operation. Give full marks only if the values after every merge operation and the number of calls to merge are correct.

   After merge : 1
   28 43 72 79 23 70 55 39 68 1 41 40 5 25 95 4 42 54 79 55

   After merge : 2
   28 43 72 79 23 70 55 39 68 1 41 40 5 25 95 4 42 54 79 55

   After merge : 3
   28 43 72 23 79 70 55 39 68 1 41 40 5 25 95 4 42 54 79 55

   After merge : 4
   23 28 43 72 79 70 55 39 68 1 41 40 5 25 95 4 42 54 79 55

   After merge : 5
   23 28 43 72 79 55 70 39 68 1 41 40 5 25 95 4 42 54 79 55

After merge : 6
23 28 43 72 79 39 55 70 68 1 41 40 5 25 95 4 42 54 79 55

After merge : 7
23 28 43 72 79 39 55 70 1 68 41 40 5 25 95 4 42 54 79 55

After merge : 8
23 28 43 72 79 1 39 55 68 70 41 40 5 25 95 4 42 54 79 55

After merge : 9
1 23 28 39 43 55 68 70 72 79 41 40 5 25 95 4 42 54 79 55

After merge : 10
1 23 28 39 43 55 68 70 72 79 40 41 5 25 95 4 42 54 79 55

After merge : 11
1 23 28 39 43 55 68 70 72 79 5 40 41 25 95 4 42 54 79 55

After merge : 12
1 23 28 39 43 55 68 70 72 79 5 40 41 25 95 4 42 54 79 55

After merge : 13
1 23 28 39 43 55 68 70 72 79 5 25 40 41 95 4 42 54 79 55

After merge : 14
1 23 28 39 43 55 68 70 72 79 5 25 40 41 95 4 42 54 79 55

After merge : 15
1 23 28 39 43 55 68 70 72 79 5 25 40 41 95 4 42 54 79 55

After merge : 16
1 23 28 39 43 55 68 70 72 79 5 25 40 41 95 4 42 54 55 79

After merge : 17
1 23 28 39 43 55 68 70 72 79 5 25 40 41 95 4 42 54 55 79

After merge : 18
1 23 28 39 43 55 68 70 72 79 4 5 25 40 41 42 54 55 79 95

After merge : 19
1 4 5 23 25 28 39 40 41 42 43 54 55 55 68 70 72 79 79 95

5. [10 points] Let's say the partition algorithm always picks the first element as the pivot. How many times will the partition algorithm be called to sort the following sequence of integer elements using quick sort?

```
28 43 72 79 23 70 55 39 68 1 41 40 5 25 95 4 42 54 79 55
```

What would be the contents of the array every time the partition procedure returns? If the partition algorithm is called k times, you need to write the contents of the whole array k times.

The partition is called 14 times.

0.5 points for the correct values of the array after a partition.
Give full marks only if the values after every partition and the number of calls to partition are correct.

After partition : 1
1 4 25 5 23 28 55 39 68 70 41 40 79 72 95 43 42 54 79 55

After partition : 2
1 4 25 5 23 28 55 39 68 70 41 40 79 72 95 43 42 54 79 55

After partition : 3
1 4 25 5 23 28 55 39 68 70 41 40 79 72 95 43 42 54 79 55

After partition : 4
1 4 23 5 25 28 55 39 68 70 41 40 79 72 95 43 42 54 79 55

After partition : 5
1 4 5 23 25 28 55 39 68 70 41 40 79 72 95 43 42 54 79 55

After partition : 6
1 4 5 23 25 28 43 39 55 54 41 40 42 55 95 72 79 70 79 68

After partition : 7
1 4 5 23 25 28 41 39 42 40 43 54 55 55 95 72 79 70 79 68

After partition : 8
1 4 5 23 25 28 40 39 41 42 43 54 55 55 95 72 79 70 79 68

After partition : 9
1 4 5 23 25 28 39 40 41 42 43 54 55 55 95 72 79 70 79 68

After partition : 10

1 4 5 23 25 28 39 40 41 42 43 54 55 55 95 72 79 70 79 68

After partition : 11
1 4 5 23 25 28 39 40 41 42 43 54 55 55 68 72 79 70 79 95

After partition : 12
1 4 5 23 25 28 39 40 41 42 43 54 55 55 68 72 79 70 79 95

After partition : 13
1 4 5 23 25 28 39 40 41 42 43 54 55 55 68 70 72 79 79 95

After partition : 14
1 4 5 23 25 28 39 40 41 42 43 54 55 55 68 70 72 79 79 95

6. [10 points] Solve the following recurrence relation using the expansion method.
$T(n) = T(n-1) + (c * n^2)$

T(n) = T(n-1) + c * n^2
= T(n-2) + c * $(n-1)^2$ + c * n^2
= T(n-3) + c * $(n-2)^2$ + c * $(n-1)^2$ + c * n^2
After kth iteration
T(n) = T(n-k) + c (n^2 + $(n-1)^2$ + $(n-2)^2$ + …+ $(n-k+1)^2$)

Give full marks if the solution up to this point is correct. Otherwise give zero.

Substituting n-k = 1
T(n) = T(1) + c* (2^2+3^2+ …+ n^2)
= T(1) + c * (1^2+2^2+3^2+ …+ n^2 - 1)
= T(1) + c * ((n * (n+1) * (2n+1))/6 - 1)