CS 577 (Epic) # Assignment 5 – Divide and Conquer Spring 2021

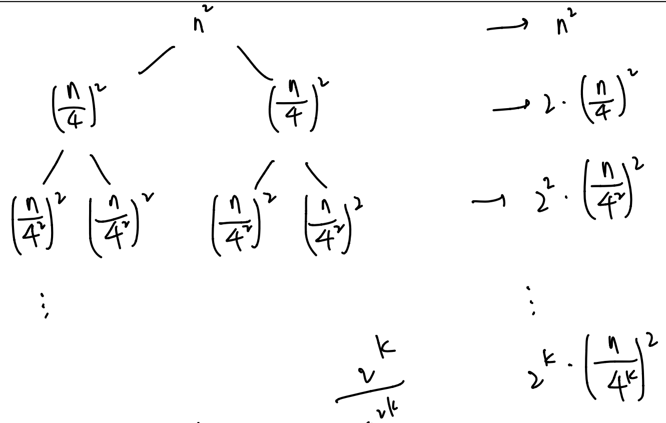
Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document.

Name: Zhuoyan Xu Wisc id: zxu444

Divide and Conquer

1. *Erickson, Jeff. Algorithms (p.49, q. 6).* Use recursion trees to solve each of the following recurrences.

   (a) $C(n) = 2C(n/4) + n^2$



n^2
$\left(\frac{n}{4}\right)^2$ $\left(\frac{n}{4}\right)^2$
$\left(\frac{n}{4^2}\right)^2$ $\left(\frac{n}{4^2}\right)^2$ $\left(\frac{n}{4^2}\right)^2$ $\left(\frac{n}{4^2}\right)^2$
$\rightarrow n^2$
$\rightarrow 2\cdot\left(\frac{n}{4}\right)^2$
$\rightarrow 2^2\cdot\left(\frac{n}{4^2}\right)^2$
$2^k\cdot\left(\frac{n}{4^k}\right)^2$
$\frac{2^k}{4^{2k}}$

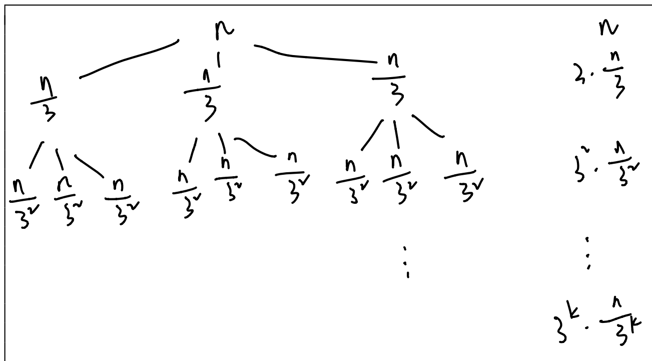
$k = \lg_4 n$

$\sum_{i=0}^{k} 2^k \cdot \left(\frac{n}{4^k}\right)^2 = n^2 \sum_{i=0}^{k} \left(\frac{1}{8}\right)^k = n^2 \cdot \frac{1-\left(\frac{1}{8}\right)^k}{\frac{7}{8}}$

$= O\left(n^2 - \left(\frac{1}{8}\right)^{\lg_4 n}\right)$

$= O(n^2)$

   (b) $E(n) = 3E(n/3) + n$



n
$\frac{n}{3}$ $\frac{n}{3}$ $\frac{n}{3}$
$\frac{n}{3^2}$ $\frac{n}{3^2}$ $\frac{n}{3^2}$ $\frac{n}{3^2}$ $\frac{n}{3^2}$ $\frac{n}{3^2}$ $\frac{n}{3^2}$ $\frac{n}{3^2}$ $\frac{n}{3^2}$
n
$3\cdot\frac{n}{3}$
$3^2\cdot\frac{n}{3^2}$
$3^k\cdot\frac{n}{3^k}$

$k = \lg_3 n$

$\sum_{i=0}^{k} 3^k \cdot \frac{n}{3^k} = n\cdot(k+1) = n \cdot \lg_3 n = n \cdot \frac{\log n}{\log 3} = O(n \log n)$

2. *Kleinberg, Jon. Algorithm Design (p. 246, q. 1).* You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values—so there are $2n$ values total—and you may assume that no two values are the same. You'd like to determine the median of this set of $2n$ values, which we will define here to be the nth smallest value.

   However, the only way you can access these values is through queries to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the kth smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

   Give an algorithm that finds the median value using at most $O(\log n)$ queries.

```
getMedian
Input: 2 equal size $\overset{\text{sorted}}{\text{array}}$. A.B. size n
Output: median
If n=1. return mean of 2 array
If n=2. A(a1, a2). B(b1, b2).
    return $\frac{1}{2}[\min(a_2, b_2) + \max(a_1, b_1)]$
let mid = $\lfloor n/2 \rfloor$
    m1 = median(A) ; m2 = median(B)
if m1 > m2: return getMedian( first-half of A, back-half of B , mid)
  else: return getMedian( back-half of A, first-half of B , mid)
```

3. *Kleinberg, Jon. Algorithm Design (p. 246, q. 2).* Recall the problem of finding the number of inversions. As in the text, we are given a sequence of n numbers $a_1, ..., a_n$, which we assume are all distinct, and we define an inversion to be a pair $i < j$ such that $a_i > a_j$.

   We motivated the problem of counting inversions as a good measure of how different two orderings are. However, this measure is very sensitive. Let's call a pair a significant inversion if $i < j$ and $a_i > 2a_j$. Give an $O(n \log n)$ algorithm to count the number of significant inversions between two orderings.

```
  Merge Count:
Input: 2 ordered List A.B
Output: merged list , count of significant inversions
Initialize S, S' to empty list. C=0. A'=A. B'=B
While either A or B or A' or B' is not empty do
    Pop and append min{front of A, front of B} to S
    pop and append min{2x front of A', front of B'} to S'
    if append item from B' to S'
        C = C + |A'|
    end
end
Return (S, C)
```

```
Count Sort
Input: List A
Output. Sorted array and number of inversions.
if |A|=1 return (A, 0)
(A1, C1) = Count Sort (first half of A)
(A2, C2) = Count Sort ( Back-half of A)
(A, C) = merge Count (A1, A2)
return (A1, C+C1+C2)
```

2 3 7 8 / 1 4 5 6

2 1 5 7 8

3+

4. *Kleinberg, Jon. Algorithm Design (p. 246, q. 3).* You're consulting for a bank that's concerned about fraud detection. They have a collection of n bank cards that they've confiscated, suspecting them of being used in fraud.

   It's difficult to read the account number off a bank card directly, but the bank has an "equivalence tester" that takes two bank cards and determines whether they correspond to the same account.

   Their question is the following: among the collection of n cards, is there a set of more than $\frac{n}{2}$ of them that all correspond to the same account? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them in to the equivalence tester. Show how to decide the answer to their question with only $O(n \log n)$ invocations of the equivalence tester.

```
merge Count:
Input: 2 list A.B with majority appearance
  m_A, m_B. (type of m_A cards larger than size(A)/2)
Output: mergedList, C. majority of this list M_C
1. merge A.B into C
2. If m_A ≠ null,
       compare m_A with every cards in C
       count appearance l_A
   else l_A = 0
3. same for m_B. get l_B
4 If l_A > 1/2 size(C) : return (C, m_A)
  else if l_B > 1/2 size(C) : return (C, m_B)
  else return (C, null)
```

```
Count Majority:
Input list A
Output: majority of A
If |A| = 1, return (A, a)
  where a is only item in A
(A_1, a) = Count Majority (first half of A)
(A_2, b) = Count Majority (Back half of A)
(A, a) = mergeCount (A_1, A_2, a, b)
return a.
```

5. Implement the optimal algorithm for inversion counting in either C, C++, C#, Java, or Python. Be efficient and implement it in $O(n \log n)$ time, where n is the number of elements in the ranking.

   The input will start with an positive integer, giving the number of instances that follow. For each instance, there will be a positive integer, giving the number of elements in the ranking. A sample input is the following:

```
2
5
5 4 3 2 1
4
1 5 9 8
```

   The sample input has two instances. The first instance has 5 elements and the second has 4. For each instance, your program should output the number of inversions on a separate line. Each output line should be terminated by a newline. The correct output to the sample input would be:

```
10
1
```

1 2

2 3 7 8 1 4 5 6

```
merge ( L1, L2 ) :   // merge 2 sorted array
 index1 = 0 ;  index2 = 0 ;
 res = [] ;  add2 = false
if index1 == L1.length && index2 == L2.length :
    return res ;
while true :
      add2 = false ;
if index1 == L1.length :
    add2 = true ;
else if  index2 != L2.length && L1(index1) ≤ L2(index2) :
    add2 = true ;
if (! add2) :
    res.append ( L1(index1) ) ;
    index1 ++ ;
else :
    res.append L2(index2) ;
    index2 ++ ;
```

```
mergeSort(L) :
 bin = ⌊L.length/2⌋
L1 = L [0 : bin]
L2 = L [bin : L.length]
L1 = L1.mergeSort ;
L2 = L2.mergeSort ;
 return merge (L1, L2)
```

the complexity of mergeSort is $O(n \log n) \le O(n^2)$

8 7 2 3 5 4 6 1

7 + 6 + 1 + 1 + 2 + 1 + 1

13