CS 577 **Assignment 5 – Divide and Conquer** Spring 2023

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: Kayley Seow Wisc id: kseow

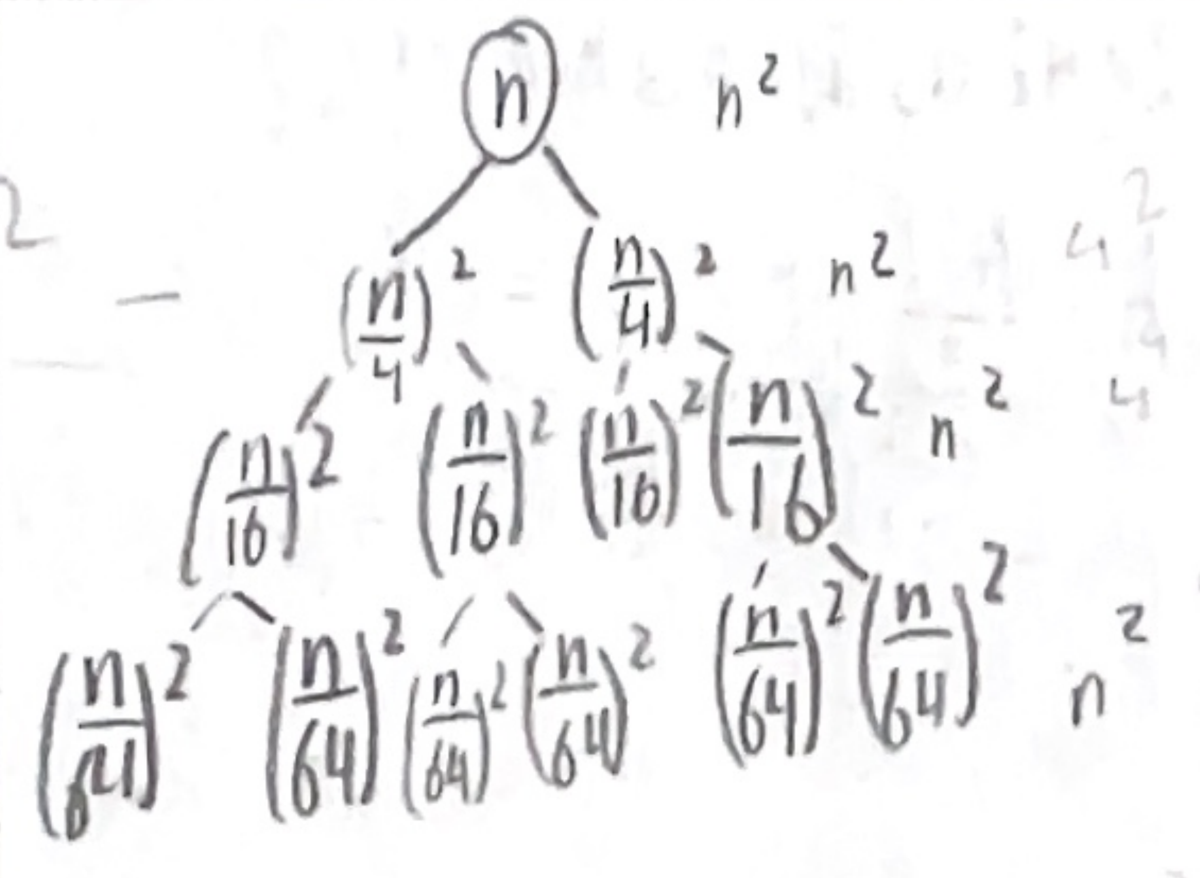
Divide and Conquer

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

   (a) $C(n) = 2C(n/4) + n^2$; $C(1) = 1$.

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

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



$\frac{n^2}{4^k} = 1$ $\quad 4^k = n^2$

$k = \log_4 n^2$

$n^2 + 2 \times (n/4)^2 + 2^2 \times \left(\frac{n}{16}\right)^2$

$+ \dots 2^{\log_4 n} \times \left(\frac{n}{4^{\log_4 n}}\right)^2$

$\left(\frac{n}{4^{\log_4 n}}\right)^2 \times 2^k$

$ar^0 + ar + ar \cdot r + ar^3 + \dots + ar^{n-1}$

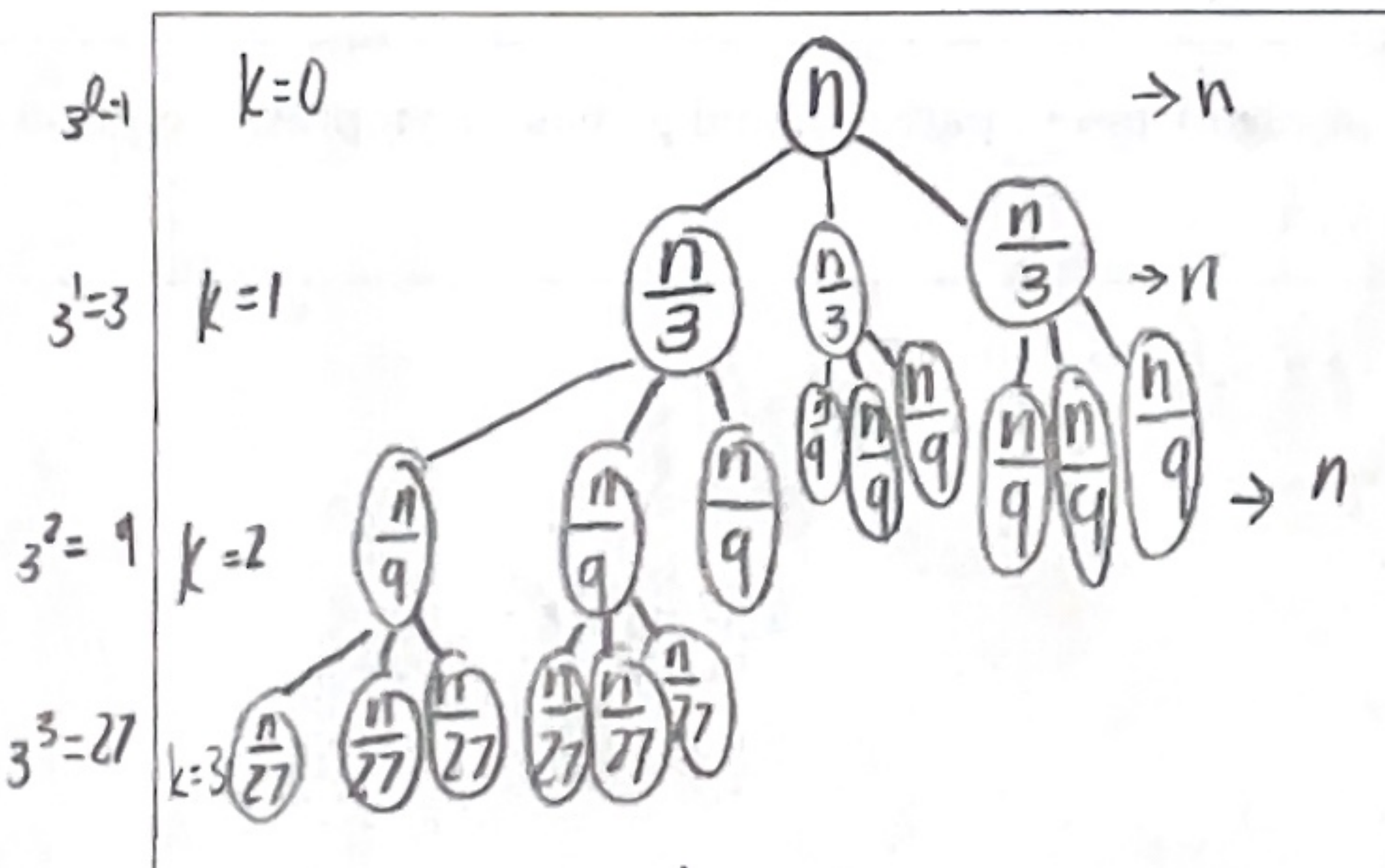
$= \frac{a \times (1 - r^{n-1})}{1 - r}$ $\quad n = \log_4 n$, $r = \frac{2}{4^2}$, $a = n^2$

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

$n^2 + n^2 \times \frac{2}{4^2} + n^2 \times \left(\frac{2}{4^2}\right)^2 + n^2 \times \left(\frac{2}{4^2}\right)^3$

$n^2 \sum_{i=0}^{\infty} \left(\frac{1}{8}\right)^i = \left(\frac{1}{1-\frac{1}{8}}\right) n^2 = \boxed{O(n^2)}$

   (b) $E(n) = 3E(n/3) + n$; $E(1) = 1$.



$\frac{n}{3^k} = 1$ $\quad 3^k = n$

$k = \log_3 n$

$n + 3\left(\frac{n}{3}\right) + 9\left(\frac{n}{3^2}\right) + \dots + \left(\frac{n}{k-1}\right)$

$n(1 + 1 + 1 + \dots + 1)$

$n(\log_3 n) + 3^k$

$\boxed{O(n \log_3 n)}$

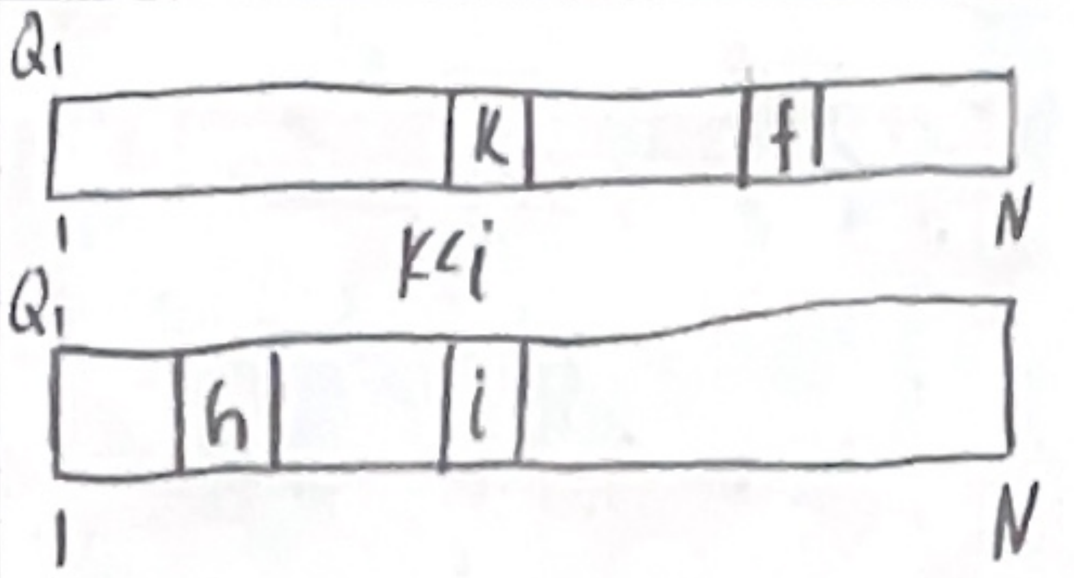
$T\left(\frac{n}{3^k}\right) = T(1)$

$\log_3 n = k$

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.

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



We set L1,H1 as low and high of Q1, and L2,H2 as low and high of Q2.

$k = \frac{N-1}{2}$ $\quad i = \frac{N-1}{2}$

base case: N of Q1, Q2 = 0

if $k > i$
  L1 = 1
  H1 = K
  L2 = N-i/2
  H2 = N
if $k < i$
  L1 = N+K/2
  H1 = N
  L2 = 1
  H2 = K

we recurse on the new portions on the query until they are

   (b) Give a recurrence for the runtime of your algorithm in part (a), and give an asymptotic solution to this recurrence.

$T(n) = T(\frac{n}{2}) + c \qquad T(1) = c$

$= T(\frac{n}{4}) + c + c$

$= T(\frac{2}{2^k}) + kc$

$\frac{n}{2^k} = 1$

$k = \log n$

$\leq c + c\log n = c(1 + \log n) \in O(\log n)$

(c) Prove correctness of your algorithm in part (a).

P(n): Median between arrays is nanimal base case n=1, array of size1 are median between them is the average.
Inductive Hypothesis: Assume P(k) holds for 1≤k≤j. Recurse each time, get P(2k), and P(k) holds and P($\frac{k}{2}$) holds. we can say that when we split array in half, have a median.
By inductive hypothesis P(2k) holds.
We know algorithm is terminate since input size is reduced to half each time.

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 $a_i > 2a_j$.

(a) Give an $O(n \log n)$ algorithm to count the number of significant inversions between two orderings.

Split array into left and right until size one, and take arrays and merge by taking smallest value of right and left array. ans while right is less than half of value compared to left array increment inversions by size of array.

(b) Give a recurrence for the runtime of your algorithm in part (a), and give an asymptotic solution to this recurrence.

$T(n) \leq 2T\left(\frac{n}{2}\right)+cn$, $T(1)=c$

$= 2\left[2T\left(\frac{n}{4}\right)+\frac{cn}{2}\right]+cn$ $\qquad 1=\frac{n}{2^k}$

$= 2\left(2\left[2T\left(\frac{n}{2^3}\right)+\frac{cn}{2^2}\right]+\frac{cn}{2}\right)+cn$ $\qquad 2^k=n$

$= 2^k T\left(\frac{n}{2^k}\right)+kcn$ $\qquad \log_2 n=k$

$= n\,T(1)+cn\log_2 n$

$= cn+cn\log_2 n = O(n\log n)$

(c) Prove correctness of your algorithm in part (a).

p(n) on input arrays size n, we return # of significant inversions.
Base Case: n=1 If our right array has value less than half the value in the left array, we have significant inversion.
Inductive hypothesis: P(k) holds for $1 \leq k < j$
whenever we perform the merge we know that we will have correct count for the prev case which is $P(\frac{k}{2})$, so when we go through array, we account for # of significant inversions between the sorted arrays. each level we will count, for number of significant inversions, so total sum is correct number and algorithm terminates bc input is half.

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.

   (a) Give an algorithm to decide the answer to their question with only $O(n \log n)$ invocations of the equivalence tester.

Divide into half until sets of 2. Will merge to check to see if we have $\frac{n}{2}$ cards tied to same account. to check the cards are of the same account. To check cards are of same account without comparing each card in each half and take card in other half, we take returned value for greatest number of cards total to same account from previous merge. Then in case of other element in the other array. i.e. half array is one value, other half is another, easily check.

   (b) Give a recurrence for the runtime of your algorithm in part (a), and give an asymptotic solution to this recurrence.

$T(n) \leq 2T(\frac{n}{2}) + n\log n$

$= 2[2T(\frac{n}{4}) + \frac{n}{2}\log\frac{n}{2}] + n\log n$ // $\frac{n}{2^k} \Rightarrow k = \log_2 n$

$= 2^k T(\frac{n}{2^k}) + kn\log n$

$= n(T(1)) + \log_2 n \cdot n\log n = cn + n\log_2 n \cdot \log n$

$= O(n\log^2 n)$

(c) Prove correctness of your algorithm in part (a).

P(n) with an input size n our program returns correct output. Base case n=1, array will be 1, will return both. Both are $\frac{n}{2}$, in case that returning 2 values we recurse there is size of $\frac{n}{2}$.
Inductive hypothesis: P(k) holds for 1≤k≤j. By inductive hypothesis, we know previous recursive step. $P(\frac{k}{2})$, must hold. Prove previous step holds, return to $\frac{(k/2)}{2}$ majority value, or none at two values each center, take returned values and compare to other input array, giving us value or not. algo terminates because each input is half size, no way to divide infinitely when smallest case is integer greater than 0.

5. Implement the optimal algorithm for inversion counting in either C, C++, C#, Java, Python, or Rust. 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
```