

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: Tung Sung ParkWisc id: 908 022 369

## Divide and Conquer

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

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

	<p>overall cost</p> $n^2$ $2 \cdot \frac{n^2}{4}$ $2^2 \cdot \frac{n^2}{4^2}$ $\vdots$ $2^k \cdot \frac{n^2}{4^k} = \frac{n^2}{2^k}$	$\sum_{i=0}^k \frac{n^2}{2^i} = n^2 \sum_{i=0}^k \left(\frac{1}{2}\right)^i$ $= n^2 \frac{1 - (\frac{1}{2})^{k+1}}{\frac{1}{2}} = 2n^2(1 - \frac{1}{2^{k+1}})$ <p><math>\frac{n^2}{2^k}</math> is the last level  <math>\Rightarrow \frac{n^2}{2^k} = 1 \Rightarrow k = \log_2 n^2</math></p> $\sum_{i=0}^k 2n^2(1 - \frac{1}{2^{k+1}}) = 2n^2(1 - \frac{1}{2^{k+1}})$ $= 2n^2 \cdot 2 = O(n^2)$
--	--	--

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

	<p>overall cost</p> $n$ $3 \cdot \frac{n}{3}$ $3^2 \cdot \frac{n}{3^2}$ $\vdots$ $3^k \cdot \frac{n}{3^k}$	$\sum_{i=0}^k (3^i \cdot \frac{n}{3^i}) = (k+1)n$ <p><math>\frac{n}{3^k}</math> is the last level of the tree <math>\Rightarrow \frac{n}{3^k} = 1 \Rightarrow k = \log_3 n</math></p> $\sum_{i=0}^k (k+1)n = O(n \log n)$
--	--	---

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  $n$ th 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  $k$ th 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.

```

FindMedian
Input  data A, data B (Both are size  $n$ , sorted in ascending order)
Output median of a combined data A and B
 $P_1 = P_2 = n/2$ 
for  $i = 2$  to  $\log n$ 
     $m_1 = \text{Query}(A, P_1)$ 
     $m_2 = \text{Query}(B, P_2)$ 
    if  $m_1 > m_2$ 
         $P_1 = P_1 - n/2^i$ 
         $P_2 = P_2 + n/2^i$ 
    else
         $P_1 = P_1 + n/2^i$ 
         $P_2 = P_2 - n/2^i$ 
end for
Return  $\min(m_1, m_2)$ 

```

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  $a_1, \dots, a_n$ , 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.

<pre> Input  an array A, left index, right index Output # inversions Divide if left &gt; right     return end <math>m = (\text{left} + \text{right}) / 2</math> <math>A = \text{Divide}(A, \text{left}, m)</math> <math>B = \text{Divide}(A, m+1, \text{right})</math> <math>C = \text{Merge}(A+B, A, \text{left}, m, \text{right})</math> Return <math>A+B+C</math> </pre>	<pre> Input  an array A, left index, middle index, right index Output # inversions Merge Count <math>P_1 = \text{left}, P_2 = \text{middle} + 1</math> (two pointers from left and middle) while <math>P_1 \leq \text{middle}</math> and <math>P_2 \leq \text{right}</math>     if <math>A[P_1] &lt; 2A[P_2]</math>         <math>P_1++</math>     else         increment # inversions by 1         <math>P_2++</math>     end end Return # inversions </pre>
---	---

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.

```

Input  an array A of cards, n
Output true if a set has more than n/2

Find
  if size of A is 1
  | return false
  end
  pivot = A[0]
  for a in A
  | if a == pivot
  | | Set1.append(a)
  | else
  | | Set2.append(a)
  | end
  end
  if size of Set1 >= n/2
  | return true
  else
  | find(Set2, n)
  end

```

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 a 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

```