Jake Traut

103134737

CSCI 3104 Algorithms Homework 3

1. Suppose you are choosing between the following three algorithms:
    a. Algorithm A solves problems by dividing them into four subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
    b. Algorithm B solves problems of size n by recursively solving two subproblems of size n − 1 and then combining the solutions in constant time.
    c. Algorithm C solves problems of size n by dividing them into nine subproblems of size n/3, recursively solving each subproblem, and then combining the solutions in O(n 2) time.
    a. Algorithm A is using the Master theorem with a=4, b=2, d=1. Because d < logb a the run time is O(n^logb a) = O(n^log2 4) = O(n^2)
    b. Algorithm B runtime can be expressed as T(n) = 2T(n-1) + C, which can be expanded as C * summation [i = 0, n-1] 2^i + 2^n * T(0) = O(2^n)
    c. Algorithm C also uses Master theorem with a=9, b=3, d=2. Because d=logb a the run time is O(n^d * logn) = O((n^2) * logn)

    I would choose algorithm A because it has the lowest order exponent O(n^2) and is a subset of O((n^2)* logn) so it is the smallest and fastest.

2. Show that any array of integers x[1, . . . , n] can be sorted in O(n + M) time, where

$$M = \max_i x_i - \min_i x_i$$

This would be done with a counting sort algorithm. The time complexity of counting sort is O(n+M) where n is the number of elements in the array and M is the range of the array. The algorithm scans through arrays of size n and M and does a constant amount of work on each element of the arrays thus the time is O(n+M).

3. A k-way merge operation. Suppose you have k sorted arrays, each with n elements, and you want to combine them into a single sorted array of kn elements.
    a. Here is one strategy: Using the merge procedure from Chapter 2.3, merge the first two arrays, then merge in the third, then merge in the fourth, and so on. What is the time complexity of this algorithm, in terms of k and n?
    This method merges the first two arrays of size n to create a new array of size 2n, then continues on to merge that array with another of size n, to make 3n, and so on.
    Run time: (n+n) + (2n+n) + (3n+n) + … + ((k-1)n+n) = 2n+3n+4n+…+kn = O((k^2)*n)
    b. Give a more efficient solution to this problem, using divide-and-conquer
    We can combine the divide-and-conquer technique with a merge sort that will merge the first array with the second, the third with the fourth, and continue this until all have been merged to into k/2 arrays of size 2n, doing O(kn) work. Then repeat the process until we have one full array of size kn, which would take O(kn) work O(log k) times so in total

the runtime will be O(kn*log k).

4. Stock maximum profit records

   TSLA.csv Start Date: 6/22/2015 End Date: 9/22/2015
   Buy Date: 7/7/2015 End Date: 8/25/2015 Max Profit: 77.405

   GOOGL.csv Start Date: 6/22/2015 End Date: 9/22/2015
   Buy Date: 7/24/2015 End Date: 8/25/2015 Max Profit: 93.42

   AAPL.csv Start Date: 6/22/2015 End Date: 9/22/2015
   Buy Date: 7/23/2015 End Date: 8/25/2015 Max Profit: 37.98

Honor Code Pledge: "On my honor, as a University of Colorado at Boulder student, I have neither given nor received unauthorized assistance."