CS 104

LAB WEEK 11

1. Write exhaustive enumeration and bisection search-based methods to approximate the square root of a positive floating number as discussed in the lectures. Both methods receive x > 1 and epsilon < 1 as input, and returns a y such that y*y is within epsilon of x. Compare running times of both methods for various values of x and epsilon.

2. Write a recursive and an iterative method to calculate factorial of a given number. Compare running times of both methods for various inputs. They will not be same even though both methods have O(n) complexity.

3. Write a recursive and non-recursive method to calculate Fibonacci series of a given n. For recursive implementation, you can use the definition discussed in the class. For non-recursive method, Fibonacci series can be explicitly calculated by the closed-form formula below:

$$f_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

   Compare running times of both methods for various input size.

4. Write a function named genPowerset(L) that returns a list of lists that contains all possible combinations of the elements of L. For example, if L is ['a', 'b'], the powerset of L will be a list containing the lists [], ['b'], ['a'], and ['a', 'b']. Analyze the running time behavior of the function for lists of various size.

5. Write a selection sort method to sort an unsorted list into a sorted list. Basically, selection sort algorithm is defined as below. What is the complexity of the algorithm in terms of O notation? How does the running time change depending on different size input lists.

   1. Divide the list to be sorted into a sorted portion at the front (initially empty) and an unsorted portion at the end (initially the whole list).
   2. Find the smallest element in the unsorted list:

      1. Select the first element of the unsorted list as the initial candidate.
      2. Compare the candidate to each element of the unsorted list in turn, replacing the candidate with the current element if the current element is smaller.
      3. Once the end of the unsorted list is reached, the candidate is the smallest element.

3. Swap the smallest element found in the previous step with the first element in the unsorted list, thus extending the sorted list by one element.
4. Repeat the steps 2 and 3 above until only one element remains in the unsorted list.