Questions and Answers of Lab 2

Yuliang Jin 986381

1

Determine the asymptotic running time of the following procedure (an exact computation of number of basic operations is not necessary):

```
int[] arrays(int n) {
   int[] arr = new int[n];
   for(int i = 0; i < n; ++i){
        arr[i] = 1;
   }
   for(int i = 0; i < n; ++i) {
      for(int j = i; j < n; ++j){
         arr[i] += arr[j] + i + j;
        }
   }
    return arr;
}
```

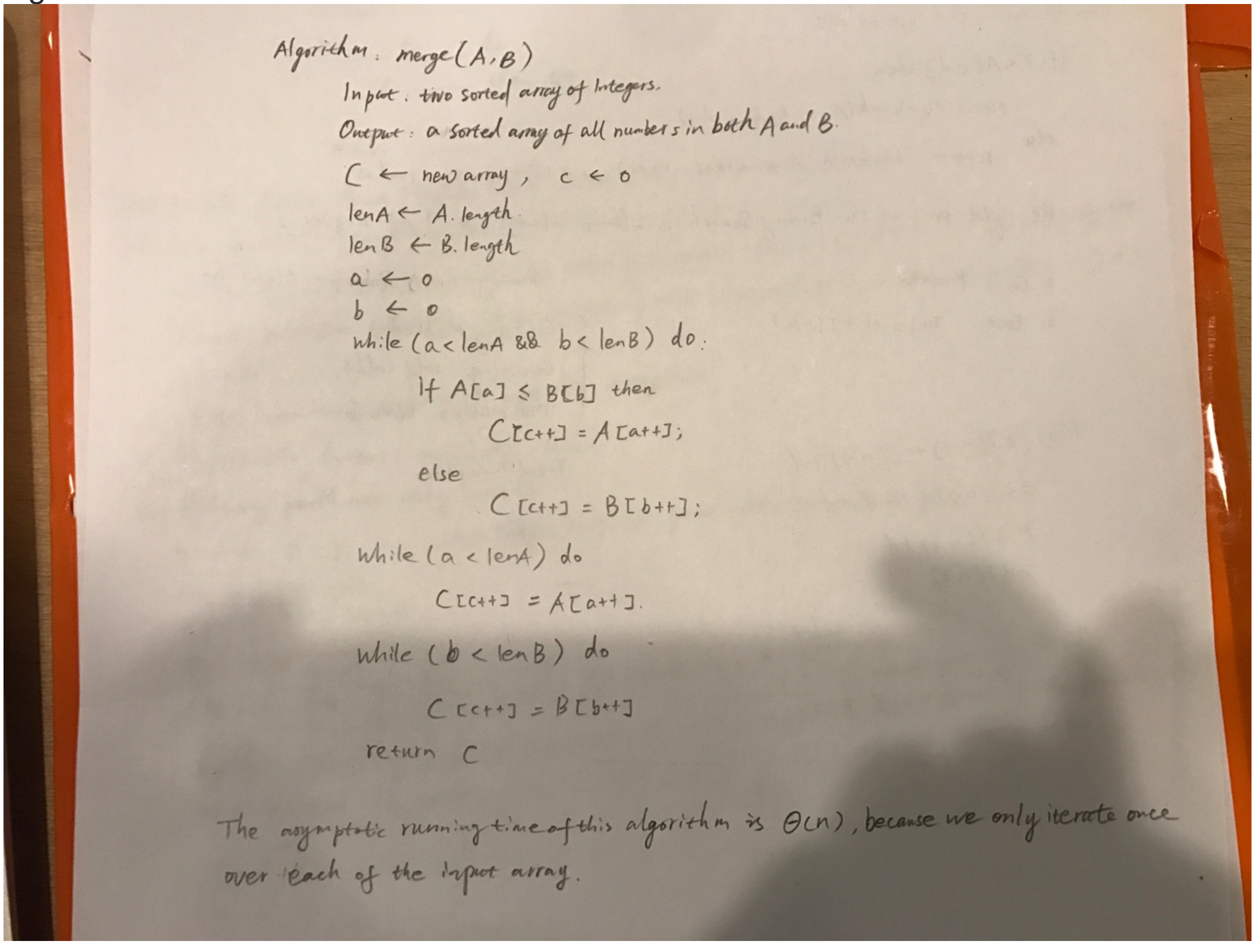
The asymptotic running time of the above procedure is $O(n^2)$.

2

Consider the following problem: As input you are given two sorted arrays of integers. Your objective is to design an algorithm that would merge the two arrays together to form a new sorted array that contains all the integers contained in the two arrays. For example, on input [1, 4, 5, 8, 17], [2, 4, 8, 11, 13, 21, 23, 25] the algorithm would output the following array: [1,2,4,4,5,8,8, 11, 13, 17, 21, 23, 25] For this problem, do the following:

1. Design an algorithm Merge to solve this problem and write your algorithm description using the pseudo-code syntax discussed in class.
2. Examining your pseudo-code, determine the asymptotic running time of this merge

algorithm



```
Algorithm: merge(A,B)
    Input: two sorted array of Integers.
    Output: a sorted array of all numbers in both A and B.
    C ← new array ,   c ← 0
    lenA ← A.length
    lenB ← B.length
    a ← 0
    b ← 0
    while (a< lenA && b< lenB) do:
        If A[a] ≤ B[b] then
            C[c++] = A[a++];
        else
            C[c++] = B[b++];
    While (a < lenA) do
        C[c++] = A[a++].
    While (b < lenB) do
        C[c++] = B[b++]
    return C
```

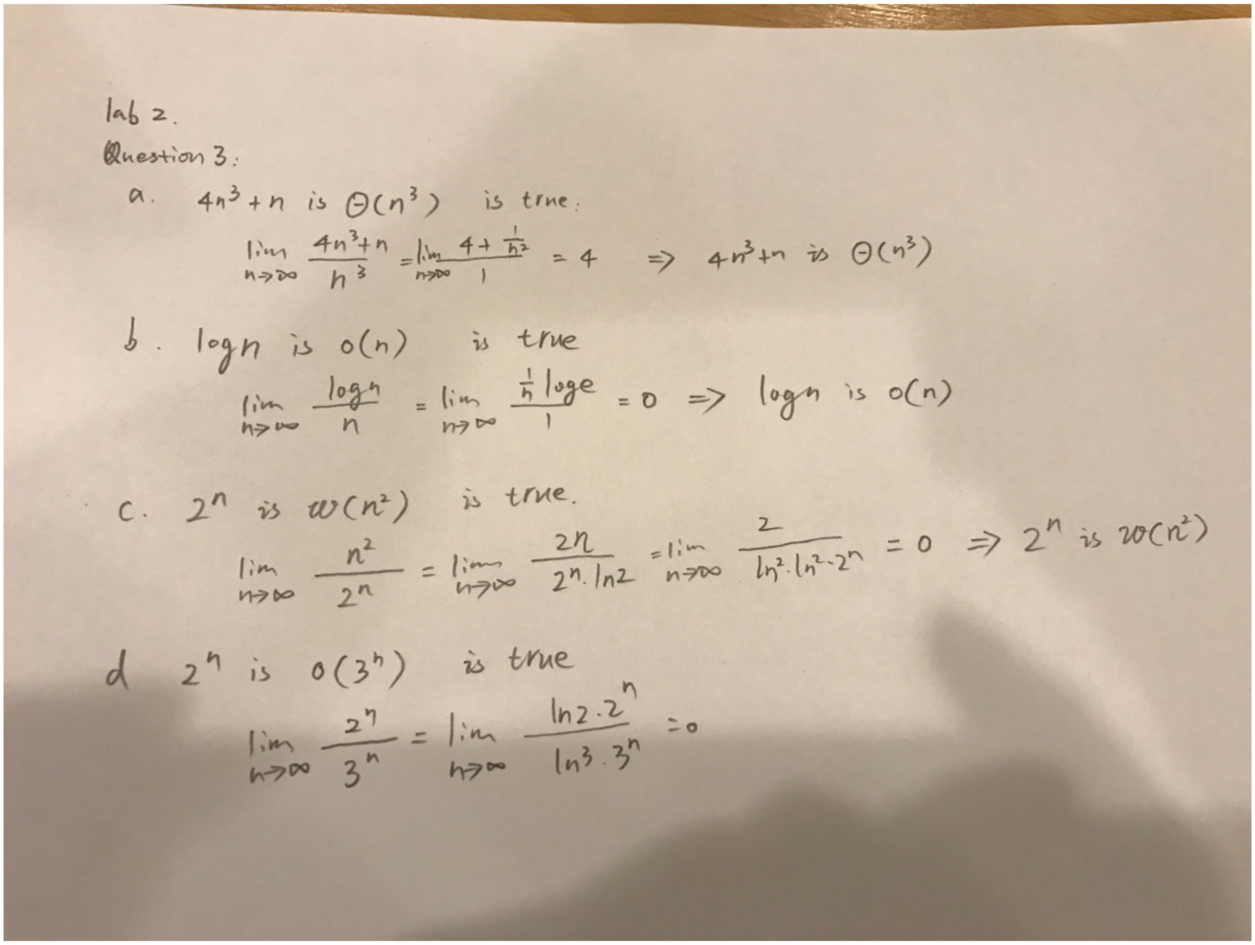
The asymptotic running time of this algorithm is $\Theta(n)$, because we only iterate once over each of the input array.

3. Implement your pseudo-code as a Java method merge having the following signature: int[] merge(int[] arr1, int[] arr2). Be sure to test your method in a main method to be sure it really works!

```
public static int[] merge(int[] arr1, int[] arr2) {
    int len1 = arr1.length;
    int len2 = arr2.length;
    int[] arr3 = new int[len1 + len2];
    int a=0, b=0, c=0;
    while(a<len1 && b<len2) {
      if(arr1[a] <= arr2[b]) {
        arr3[c++] = arr1[a++];
      } else {
        arr3[c++] = arr2[b++];
      }
    }
    while(a<len1) {
      arr3[c++] = arr1[a++];
    }
    while (b<len2) {
      arr3[c++] = arr2[b++];
    }
    return arr3;
  }
```

3

Use the limit definitions of complexity classes given in class to decide whether each of the following is true or false, and in each case, prove your answer.



lab 2.

Question 3:

a. $4n^3+n$ is $\Theta(n^3)$ is true:

$$\lim_{n\to\infty}\frac{4n^3+n}{n^3}=\lim_{n\to\infty}\frac{4+\frac{1}{n^2}}{1}=4 \Rightarrow 4n^3+n \text{ is } \Theta(n^3)$$

b. $\log n$ is $o(n)$ is true

$$\lim_{n\to\infty}\frac{\log n}{n}=\lim_{n\to\infty}\frac{\frac{1}{n}\log e}{1}=0 \Rightarrow \log n \text{ is } o(n)$$

c. 2^n is $\omega(n^2)$ is true.

$$\lim_{n\to\infty}\frac{n^2}{2^n}=\lim_{n\to\infty}\frac{2n}{2^n\cdot\ln 2}=\lim_{n\to\infty}\frac{2}{\ln 2\cdot\ln 2\cdot 2^n}=0 \Rightarrow 2^n \text{ is } \omega(n^2)$$

d 2^n is $o(3^n)$ is true

$$\lim_{n\to\infty}\frac{2^n}{3^n}=\lim_{n\to\infty}\frac{\ln 2\cdot 2^n}{\ln 3\cdot 3^n}=0$$

4 Power Set Algorithm

Given a set X, the power set of X, denoted P(X), is the set of all subsets of X. Below, you are given an algorithm for computing the power set of a given set. This algorithm is used in the brute-force solution to the SubsetSum Problem, discussed in the first lecture. Implement this algorithm in a Java method:

```
public static List<Set<Integer>> powerSet(List<Integer> X) {
    List<Set<Integer>> P = new ArrayList<Set<Integer>>();
    HashSet<Integer> S = new HashSet<Integer>();
    P.add(S);
    HashSet<Integer> T = new HashSet<Integer>();
    while(!X.isEmpty()) {
      List<Set<Integer>> temp = new ArrayList<Set<Integer>>();
      Integer f = X.remove(0);
      //When iterate over P, P should not be modified(like adding elements or delete
elements during the process of
      // iteration) at the same time, otherwise ConcurrentModificationException will
be thrown.
      for (Set<Integer> x : P) {
        T = new HashSet<Integer>(x);
        T.add(f);
        temp.add(T);
      }

      for (Set<Integer> integers : temp) {
        P.add(integers);
      }
    }
    return P;
  }
```

5

In the slides, an algorithm removeDups was given for extracting a list of all the distinct elements of a given input list L.

Explain why the running time of removeDups is O(n^2)

Because in the contains method of the ArrayList, it iterates over the list data using a for loop. Therefor, this algorithm is actually a nested for loop. So the running time is O(n^2).

Try using the technique shown in the solution to the Sum of Two problem (i.e. a HashMap) to improve running time of removeDups to O(n)

```
public static List<Integer> removeDups(List<Integer> L) {
    ArrayList<Integer> list = new ArrayList<Integer>();
    HashMap<Integer, Integer> map = new HashMap<Integer, Integer>();
    for (Integer integer : L) {
      if(!map.containsKey(integer)) {
        map.put(integer, 1);
        list.add(integer);
      }
    }
    return list;
  }
```