Assignment 1

Bioinformatics (CIS 455)

Javier Arechalde

February 6, 2018

Problem 1-1 Jones & Pevzner, Problem 2.1

1.

We are going to write the pseudocode for an algorithm, that given a list of n numbers, returns the largest and smallest number in that list.

Algorithm 1 Finding largest and smallest numbers

```
1: Initialize $max$ to $L[0]$ and $min$ to $L[0]$
2: for $n$ in number list L do
3:     if $n < min$ then
4:         $min = n$
5:     end if
6:     if $n > max$ then
7:         $max = n$
8:     end if
9: end for
10: Return $max$ and $min$
```

The time complexity of my algorithm will be $O(n) = n$, and the running time will be $2(n-1)+2$ because for every number in the list that starts from the second position after initialization, we need to check if the number is greater than the current maximum, or smaller than the current minimim.

2.

Now we will implement an algorithm that performs only $3n/2$ comparisons to find the smallest and largest numbers in the list. For this implementation we will use the tournament method described below (Divide & Conquer).

Algorithm 2 Divide & Conquer pseudocode

```
 1: We have a list $l$ with $n$ integers $l = (l_1, l_2, ...l_n)$
 2: function FINDMAXMIN(list,alow,ahigh)
 3:     if $lengthlist = 1$ then
 4:         Maximum and minimum are the same
          return maximum and minimum
 5:     end if
 6:     if $lengthlist = 2$ then
 7:         Compare both elements in the list to find the maximum and minimum
          return maximum and minimum
 8:     end if
 9:     Divide the array in half, and store the two halves in arrL and arrR
10:     Call findmaxmin to find the maximum and minimum of arrL and arrR
11:     if Maximum of arrL or arrR is greater than the current max then
12:         Update global maximum
13:     end if
14:     if Minimum of arrL or arrR is smaller than the current min then
15:         Update global minimum
16:     end ifreturn the global maximum and the global minimum
17: end function
18: In the end, we have the maximum and minimum of the given list of integers.
```

In this case, we will divide the initial list with n numbers into two lists of $n/2$ each, for eacg of this lists, we will need to find the maximum and minimum of this lists, which can be done in $n/2 - 1$ comparisons. So in total we will need $3n/2 - 2$ comparisons to find the maximum and minimum of a list containing n numbers.

Problem 1-2 Jones & Pevzner, Problem 2.2

Now we will write the pseudocode for two algorithms that iterate over every index from $(0, 0, ..., 0)$ to $(n_1, n_2, ..., n_d)$, one of them will be recursive and the other one iterative.

Iterative pseudocode:

Algorithm 3 Iterative pseudocode

```
We have two indexes lists $l = (0, 0, ...0)$ and $n = (n_1, n_2, ...n_d)$.
We will print all the possible combination of elements
**for each** $(i_1, i_2, .., i_d)$ **from** $(0, 0, ..., 0)$ **to** $(n_1, n_2, ..., n_d)$ **do**
    Return list $i$.
**end for**
```

Recursive pseudocode:

Algorithm 4 Recursive pseudocode

```
We have two indexes lists $l = (0, 0, ...0)$ and $n = (n_1, n_2, ...n_d)$.
We will recursively build a tree to print all the possible combinations
**function** COMBINATIONS(n,lev,index)
    **while** lev¡d **do**
        Add n$_i ndextolist$list        **if** $level = 0$ **then**
        Return array of 0's

    **if** index = d **then**
        Reset index
    **end if**
    **if** We reach the lenght of l and used all d **then**
        Go down one level and reset index
    **end if**
    **for** $n_i$ in $n$ **do**
        Return () COMBINATIONS(n,lev,i)

        Return the *sum* value
```

Problem 1-3 Jones & Pevzner, Problem 2.3

- Yes, $\log n = O(n)$, because $O(n)$ is an upper bound for $\log n$, as it grows faster.
- No, $\log n = \Omega(n)$, because $\Omega(n)$ is not a lower bound for $\log n$, as $\log n$ grows significantly slower.
- No, $\log n = \Theta(n)$, because $\log n$ is not $O(n)$ and $\Omega(n)$ at the same time.

Problem 1-4 Jones & Pevzner, Problem 2.17

Will the viruses eventually kill all the bacteria?

In order to find if the viruses end up killing all the bacteria in the Petri dish, we only need to prove if the growth of the viruses is higher than the growth of the bacteria.

In the first minute, the virus kills one bacteria, and produces another copy of himself, and all the remaining bacteria reproduce, making 2 viruses and $2(n-1)$ bacteria.

Then the viruses number will continue to double each step, and the number of bacteria will be the number of initial bacteria minus the bacteria that is killed by every virus at each step, and then doubled. The key here is that the virus continue to double at each step, and even though the number of bacteria doubles at each step too, this number is reduced by the number of viruses, before doubling, so we can say that the rate of growth of the viruses is higher than the bacteria growth rate, and at some point they will be exterminated by the viruses.

Algorithm design

Algorithm 5 Algorithm for calculating the number of steps

```
1: At the beginning we have $n_v = 1$ virus and $n_b = n$ bacteria
2: At each step, the number of viruses double, the bacteria double too, but $n_v$
   are killed by the virus too before they double.
3: while $n_b > n_b$ do
4:     $n_b = 2 * (n_b - n_v)$
5:     $n_v = 2 * n_v$
6: end while=0
```