HOMEWOR #4 Mustafa Tokgöz
171044077

Answer of Question 1)

n zero text, 0010

Algorithm makes 2 successful compare and 1 unsuccessful compare becouse of input 1. So number of character comparision is $3 \cdot n = \boxed{3n}$

For the worst case of input 3-bit pattern, worst input is 001 in 3-bit. So the worst case $\boxed{O(3n)}$

Answer of Question 2)

. A to E = 3 , E to B = 1 , B to C = 6 , C to D = 2

3 + 1 + 6 + 2 = 12 .

Upper side is an example of applying brute-force algorithm.

There is a lot of road to reach every city.

The answer is = A to D = 4 , D to C = 2, C to E = 4,

E to B = 1

$\boxed{4+2+4+1 = 11}$ the smallest reached number

Answer of Question 3)

Algorithm - Q3 (n):

```
        if n == 1
            return 0

        else
            return 1+ Algorithm - Q3 ( floor(n/2))

end
```

Time complexity of this algorithm =

$$T(n) = \begin{cases} 0 & n = 1 \\ 1+T(floor(n/2)) & n > 1 \end{cases}$$

If we use Master Teorem

$a = 1$ $b = 2$ $d = 0$

$a = b^d$

$1 = 2^0$ then $\Theta(n^d \log n)$

So Tim complexity is $\Theta(n^0 \log n) = \boxed{\Theta(\log n)}$

Answer of Questien 4)

```
Algorithm-Q4 (array bottles):

        while length of bottles >1:
                n= length of bottles/2
                L1= bottles[0,1,----,n-1]
                L2= botteles [n,-----,2n-1]

                 if ( weight of L1 < weight of
                       L2)
                         check=1
                 end if
                 else if(weight of L1> weight
                       of L2)
                              check=-1
                   end if
                   else
                          check = 0
                   end else

                    if (check ==1)
                            bottle = L1
                     end if
                     else if (check==-1)
                             bottle=L2
                     end if
                     else if (Check ==0)
                            L3 = bottle [2n]
                            bottle= L3
                     end if
end while
    return bottle[0]
end function
```

the Time Complexity for this algorithm=

The best Case= $\boxed{\Theta(1)}$ → if length of bottle = 1

The Avarage Case and the worst Case = $\boxed{\Theta(\log_3 n)}$

Because if we use Master teorem

$T(n) = T(n/3) + 6$

$a=1 \quad b=3 \quad d=0$

$a = b^d \Rightarrow 1 = 3^0 \Rightarrow \Theta(n^d \cdot \log_3 n) = \boxed{\Theta(\log_3 n)}$

Explanation:

In my algorithm, I split the bottles 0 to length/2 and
length/2 to length. Then I check weight of them.
To be incorrect bottle, it must be less weight than
the others becouse There is no way to fill the bottle
more weight than its volume. So I compare them and
if weight of first one is less than the other
then, result in the first one. if more than the
other one then the result in the second one.
if equals, then result is the last one. These
all in while part. It loops until one lasts.
Then it returns first element of the list and
this is result.

Answer of Question 5)

```
h_algorithm_ Question5(array1 , n, array2, m, x):
        if ( x > n+m)
            return -1   // fail
        end if
        if ( x ≤ 0)
            return -1
        end if
        if( n>m)
            return h-algorithm-Question5(array2, m, array1,
                                          n, x)
        end if
        if(n==0)
            return array2[x-1]
        end if
        if(x == 1)
            if (array[0] < array2[0])
                return array1[0]
            end if
            else return array2[0)
            end else
        end if
        if(m < x/2)
            i = n
        endit
        else
            i = x/2
        end else
```

```
if (m<x/2)
        J=m
  end if
   else
        J=x/2
   end else

    if (array[i-1]> array[J-1])
        return h-algorithm-Question5(array1, n, array2+J, m-J,
                                      x-J)
     end if
     else
        return Algorithm-Question5(array1+i, n-i, array2,
                                      m, x-i)
      end else
end function

Algorithm-Question5(array, array2, x):
            n= length of array
            m= length of array2
            MergeSort(array)
            MergeSort(array2)
            h-algorithm-Question5(array, n, array2, m, x)

   end function
```

The worst Case= $\overbrace{O(n\log n)}^{\text{MergeSort}} + \overbrace{O(n\log n)}^{\text{MergeSort}} + \overbrace{O(\log(n+m))}^{\text{h-algorithm-Question5}} = \boxed{O(n\log n)}$

↳ Becouse At most it can take n+m, in code first if part.