CSE 321 - Homework #4

Harun ALBAYRAK - 171044014

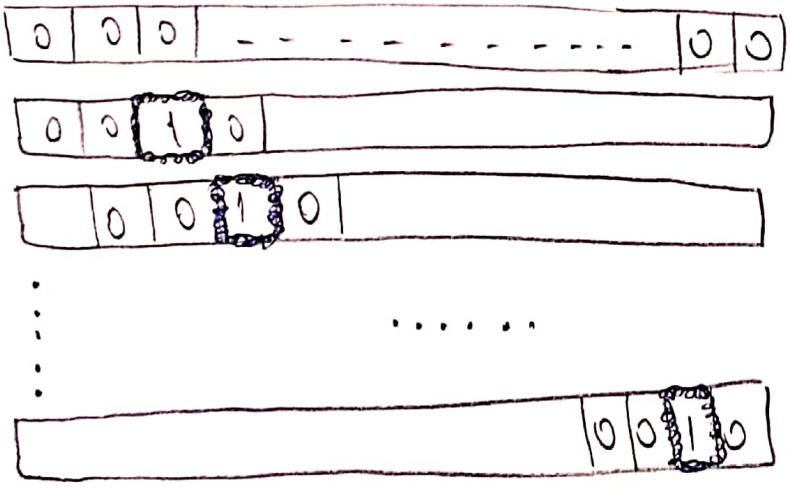
Q1)

```
procedure StringMatch(T[0,...,n-1], P[0,...,m-1])
    loop i:= 0 to n-m do
        j = 0
        while j < m and P[j] = T[i+j] do
            j = j+1
        end while
        if j == m
            return
        end if
    end loop
    return -1
end
```

$$\text{Worst case} = \sum_{i=0}^{n-m} \sum_{j=0}^{m-1} 1 = \sum_{i=0}^{n-m} m = m.(n-m+1) = m.n - m^2 + m \in O(m.n)$$

text = 000--- text length = n

pattern = 0010 pattern length = m = 4



(n-m+1) times string comparison => (n-4+1) = n-3

3 times character comparison per string comparison => 3

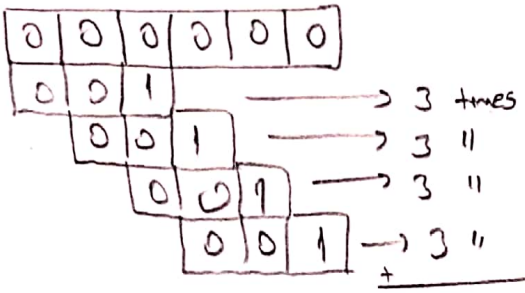
Total character comparison = 3n-9

→ Worst case the pattern should be 001.

$m=3 \Rightarrow m.(n-m+1) = 3n-9+3 = \underline{3n-6}$

→ example ⇒ text = 000000 ⇒ length = n = 5

pattern = 001 ⇒ length = m = 3



⇒ $m(n-m+1)$

$3(5+1) = \underline{12 \text{ times}}$

(Q2)

```
routes = []
procedure question2(node, letters, path, distance)
    path.append(node)
    if len(path) > 1 then
        distance += letters[path[-2]][node]
    if (len(letters) == len(path)) and (path[0] in letters[path[-1]]) then
        path.append(path[0])
        distance += letters[path[-2]][path[0]]
        routes.append([distance, path])
    for letter in letters
        if (letter not in path) and (node in letters[letter]) then
            question2(letter, dict(letters), list(path), distance)
end
---
letters = { 'A': {'B':5, 'C':5, 'D':4, 'E':3}, 'B': {'A':5, 'C':6, 'D':7, 'E':1},
            'C': {'A':5, 'B':6, 'D':2, 'E':4}  'D': {A:4, 'B':7, 'C':2, 'E':6},
            'E': {'A':3, 'B':1, 'C':4, 'D':6}}

question2('A', letters, [], 0)
routes.sort()
if (len(routes) != 0):
    print("Shortest route %s " % routes[0])
```

Q3) Algorithm to find the floor value of $\log_2 n$:

```
procedure question3 ( intNum)
    if  n ==1 :
        return 0
    else :
        return  question3 ( intNum /2 ) + 1
end
```

The reaurrence relation for the number of addition is :

$$T(n) = \begin{cases} 0 & , \text{for } n = 1 \\ 1 + T(\lfloor \frac{n}{2} \rfloor) & , \text{for } n > 1 \end{cases}$$

We can sdve with master theoren : $a = 1$, $b = 2$, $f(n) = 1$

$$n^{\log_b a} = n^{\log_2 1} \longrightarrow 0 \qquad = n^0 = 1$$

$$f(n) = \Theta(n^{\log_b a})$$

$$T(n) = \Theta(n^{\log_b a} + \log n)$$

$$T(n) = \Theta(1 + \log n)$$

$$T(n) = \Theta(\log n)$$

Question 4

Q4) Decrease and Conquer algorithm

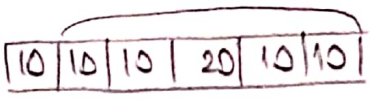
In this approach; In order to find solutions to the given problems, a solution is sought for an example of a smaller size than the problem. The solution for its smaller size is applied to the main problem.

In our problem => The weight of one of the bottles is set incorrectly.

Example => 10 10 10 20 10 10

↳ Wrong weight

The output will be 20.

my algorithm : 

i = 0, num = 0, m = 0

↳ v = array[i] = 10

* The v element is compared with the other elements of the array one by one.
* If the v element is different from the element of the array, then m is increased by 1.
* If m is equal to the number of turns in the second loop, the element v is the number that is different.
* If a diffeat number is found, the loop is exited and the value is returned by writing to the 'num' variable.
* But, if the different number is at the end, the number is found with different algorithm. If the number of turns of the second loop is 1, if the first number is not equal to the number in the loop, this is the number that is different and is returned.

My code :

```
procedure question4(array)
    flag = 0
    loop i=0 to len(array) do
        v = array[i]
        num = 0
        m = 0
        if flag == 1
            return num
        loop j=i+1 to len(array) do
            if len(array) - (i+1) == 1:
                if array[j] != array[0]:
                    flag = 1
                    num = array[j]
                    break
            if v != array[j]:
                m = m+1
            if m == len(array) - (i+1):
                flag = 1
                num = v
                break
```

The different number is found.

The different number is found.

Best case => The best case happens if the first number is different. The first loop is executed only once, and the second loop is executed for the rest of the array except the first. Its time complexity is $O(n)$.

Worst case => The worst case happens if the last number is different. Its time complexity is $O(n^2)$.

Average case => $O(n^2)$

Q5) I used the divide and conquer algorithm to solve this problem. Firstly, I sorted the each array up to the x^{th} index. Because in the worst case, it will be sufficient to look up to the lowest x number in both arrays to find the number we are looking for. I used selection sort as the sorting algorithm because selection sort brings the smallest number to the right position every time.

```
procedure selection-sort(list, n):
    loop i=1 to n-1 : do:
        min = i
        loop j=i+1 to n do:
            if list[j] < list[min] then
                min = j
            end if
        end loop
        if indexMin != i then
            swap list[min] and list[i]
        end if
    end loop
end

procedure question5(list1[x], list2[y], n):
    if x+y < n then:
        return
    a1,a2 = n
    if a1 > x then:
        a1 = x
    if a2 > y then:
        a2 = y
    selection-sort(list1, a1)
    selection-sort(list2, a2)
    return question5(list1[0:a1], list2[0:a2], n)
end
```

After sorting, we can use divide and conquer algorithm.

List3 is the sorted array of mersing list1 and list2. If middle element of list 2 < middle element of list 1 and x/2 + y/2 < k then k is between middle element of list2 and the end of arrays. If x/2 + y/2 > k it is from starts to middle element of list1 the reverse is true otherwise. We can apply the previous statement by swapping list1 and list2.

```
Procedure question5-2 ( list1[x] , list2[y] , n):
    if list1 == NULL then :
        return list2[n]
    IF list2 == NULL then :
        return list1[n]
    else :
        if x/2 + y/2 <= n then :
            if list1[ x/2] > list2[ y/2] then :
                question5 ( list1[0:x] , list2[ y/2+1:y] , n-(y/2)-1)
            else :
                question5 ( list1[ x/2+1 :x] , list2[0:y] , n-(x/2)-1)
        else :
            if list1[ x/2] > list2[ y/2] :
                question5 ( list1[0 : x/2] , list2[0: y] , n)
            else :
                question5 ( list1[0:x] , list2[0: y/2] , n)
end
```

Time complexity of question5_2 : $\Theta(\log x + \log y)$

Worst time complexity : $\Theta(nx + ny + \log x + \log y)$.
