Algorithms and Complexity
COMP_90038 Semester 2 August 2020

Assignment 1

Student Id : 1124298 Student Name: Sakshi Chandel

Question 1 :

Answer 1 :
When parameters used as pass by value then when we are changing the variable inside the function , it would not reflect or change the variable 'reference' or value outside the function. Pass by reference make sure the reference address of the variable is passed and when its value is changed ,its reflected outside the function as well.

Question 2:

Answer 2 : If buffer Is full,will remove element at readindex and if buffer is not full we will add element at writeindex. **Procedure is as follows ..**

```
procedure PUTITEM (A,C,readindex,writeindex,x)

    if (writeindex-readindex==C)
        rindex ←—— readindex % C
        A[rindex] ← x
        readindex ←— readindex + 1
        writeindex ←— writeindex + 1
    else
        windex ←— writeindex % C
        A[windex] ← x
        writeindex ←— writeindex + 1
```

Question 3 : Given n= writeindex-readindex

Answer 3 :

For the worst case scenario of the procedure:
REMOVEOLDER(A,C,readindex,writeindex,t)…,
all the elements in the ringbuffer will have time <t
Here,when n=0 i.e (readindex=writeindex) and basic operations are 2

1) **A[readindex % C].time<t**
2) **readindex ← readindex +1**

No. of basic operations for n=0 will be :

T(0) = 0 ---------- eq 0 Also,
$T(n) = 2 + T(n-1)$ ---------eq 1 , where T(n) is the time complexity of input size n
Now using telescoping (Backward Substitution),
T(n-1) = 2 + T(n-2) ------ eq 2
T(n-2) = 2+ T(n-3) --------eq 3
T(n-3) = 2 + T(n-4) --------eq 4
Using eq 3 in eq 2 ,
T(n)= 2+2+T(n-3) ----------eq 2
Using eq 4 in eq 2
T(n)= 2+2+2+T(n-4)……
When equation approaches to n…
T(n) = 2*n+ T(0)
From eq 0 , T(0) =0

Complexity :- T(n) =O(2*n)

Complexity of the function REMOVEOLDER has complexity Big O(2*N) for worst case scenario.

Question 4:

Answer 4 : Where procedure bSearch will find the smallest closest element 'a' to t present in the ring buffer ,if the closest element found is less than t then we will find the last occurrence of 'a' in the ring buffer containing duplicates of a's .If the closest element 'a' is equal to t then will find the first occurrence in the ring buffer .Procedure for removing elements having time <t in worst case scenario in complexity theta log n assuming the x2.time>x1.time (x2 added after some time x1) would be :

```
function removeElement (A,C,readindex,writeindex,t)
        a ← bSearch(A,C, readindex,writeindex,t)
        if(a=t)
            index ← firstOccurence(A,C,readindex,writeindex,a)
            readindex=index;
        if(a<t)
            index ← lastOccurence(A,C,readindex,writeindex,a)
            readindex=index+1;
```

```
function bSearch(A,C,readindex,writeindex,a)
    r ← readindex
    w ←writeindex
    if(t<A[r%C]
        return A[r%C]
    if(t>A[w-1%C]
        return A[w-1%C]
    while(r<=w-1)
        mid ← (r+w-1)/2
        if(a<A[mid%C])
            w ← mid-1;
        else if (a>A[mid%C])
            r ← mid+1;
        else
            return A[mid%C]

    return (A[r%C]-a) < (a-A[w-1%C]?A[r%C]:A[w-1%C]


function firstOccurence(A,C,readindex,writeindex,a)
    result ← -1;
    left ← readindex;
    right ← writeindex-1;
    while(left<=right)
        mid  ←  (left+right)/2
        if(a=A[mid%C])
            result ← mid;
            right ← mid-1;
        else if (a<A[mid%C])
            right ← mid-1
        else
            left ← mid+1;
    return result
```

```
function lastOccurence(A,C,readindex,writeindex,a)
        result ← -1;
        left ← readindex;
        right ← writeindex-1;
        while(left<=right)
              mid ← (left+right)/2;
               if(a=A[mid%C])
                     result ← mid;
                     left ← mid+1;
                else if (a<A[mid%C])
                      right ← mid-1
                else
                      left ← mid+1;
         return result;
```

Question 5:

Answer 5: Since log n algorithms are based on divide and conquer . So to find elements in the ring buffer if t1<=x.time<t2 .If we find all elements in the ring buffer which satisfies t1<x.time<t2 assuming x2.time>x1.time and delete all elements between the given range in log n complexity then shifting the elements added before this time frame to take empty spaces would not be possible in complexity of log n (might take O(n) complexity).

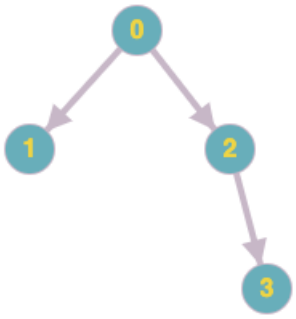
Question 6:

Assuming invariant is maintained before and after the function finishes Implementaion of the standard BFS traversal algorithm using the ring buffer as the queue of nodes to be visited is below :

```
function  BFS (⟨V,E ⟩)
      mark each node in V to 0
      readindex ← 0;
      writeindex ←0;
      C ← size(V)
      A ←init(C)
      for each v in V do
            if v is marked with 0 then
                injectVertex(A,C,readindex,writeindex,v)
                 while queue is non-empty do
                     u ← ejectVertex(A,C,readindex,writeindex,v)
                     for each edge (u,w) do
                          if w is marked with 0 then
                               injectVertex(A,C,readindex,writeindex,w)

function init(C)
       A={} (creating array of size C)
       return A

 function injectVertex(A,C,readindex,writeindex,v)
        A[writeindex%C]← v
         writeindex ← writeindex+1

function ejectVertex(A,C,readindex,writeindex)
          u=A[readindex%C]
          readindex ← readindex+1
          return u
```

Question 6a):

Answer 6 a) **No, every node will not be guaranteed to be traversed .It is a possibility that it can be .The reason being if C (Capacity of ring buffer) is less than the node then according to the algorithm if any time any node when ejected after being visited ,the ring buffer should have space greater than or equal to the number of its neighbour nodes so as to guarantee that every node of the graph is being visited successfully.**



Example – For above graph,if C is 1 (i.e it is less than number of nodes) .Its BFS is not possible.

Question 6 b):

Answer 6 b) : The solution below will tell BFS is possible sometimes even if its not possible .But wont say not possible if it is.

In the algo ,have maintained an array to find the edges of all nodes that are not visited before and while exploring that array if non visited edges of any node is less than n(the current empty space of the buffer) then BFS not possible else it is .

The modification of DFS to find that BFS is possible or not is below :

```
function DFS(<V,E>,C)
        mark each node in V with 0
        initialize neighbour of each node in V to 0 (each neighbour[v] in V to 0 )
        for each v in V do
            if V is marked with 0 then
                DFSExplore(v)
        x←true;n=C-1;
        for each v in neighbour do :
            n++
            If v has neighbour>n (if neighbour[v]>n)
                X←false
                break
            else
                n=n-neighbour[v]
        if(x←true)
            print yes
        else
            print no


function DFSExplore(v)
        count ← count+1
        mark v with count
        for each (v,w) do
            if w is marked with 0 then
                neighbour[v] ++
                DFSExplore(w)
```

Question 7 :

Answer 7 :

Given String A, B in ring buffer each of size C
Also an algorithm in O(p+s) which takes 2 string and return if string p contains s.
Let A has readindex r1,writeindex w2 and B has readindex r2,writeindex w2.
To find A,B are rotation of each other we can extract the characters of A and B in strings called string1,string2 resp. then add concatenate string1 with itself and check string1 contains string2.
Below is the pseudocode of the algorithm.

```
procedure checkRotation(A,B,r1,w1,r2,w2)
        create empty string1 of size 2*C
         create empty string2 of size C
         r1=readindex1;r2=readindex2;w1=writeindex1;
         w2=writeindex2
         for(i=r1;i<w1;i++)
              string1=string1+A[i%C]
          for(i=r2,1<w2,i++)
               string2=string2+B[i%C]
           string1=string1+string1
           answer= stringMatching(string1,string2)   (already given)
           if(answer=-1)
                  print Ring buffer are not rotation of each other
           else
                  print Ring buffer are rotation of each other
```

Submitted by:
Name : Sakshi Chandel
Student ID : 1124298