**CSE 222**

**HOMEWORK 4 REPORT**

**----------------------------------------------**

**ERSEL CELAL EREN**

**1901042673**

**1 - System Requirements**

**In this task, we are asked to write recursion functions, analyze and compute their time complexities. We use recursion to add clarity, reduce the time needed to write ,debug and read easily code. Recursion helps us to reduce the time complexity and system resource usage of programs. Proper recursion method shouldn’t have iterative ways. Also it should be considered to StackOverflowException.**

**3 - Problem Solution Approach**

**Q1 – For first question, I wrote a method that takes 2 string and 1 count integer as parameter. Then, this method calls a method with same name but 4 parameter. 4th parameter -integer count- is used to return -1 when number of occurrences is less then given i. I used ‘indexOf’ method to find first occurrence of little string and call new recursive method with remaing string after the found index by indexOf method. To pass that new string I used ‘substring’ method. After null reference check, method finds i’th occurrence and returns index of it.**

**Q2 – In second question, I found exact position for upper bound and lower bound. If we have a an sorted array like [ 0 , 3 , 4 , 6 , 9] and our lower bound is 1 and upper bound is 5, recursive method returns 2.5 for upper bound and 0.5 for lower bound. These double values are our indexes. Final result is found by subtraction upperindex – lowerindex which 2.5 – 0.5 = 2. There are 2 numbers between 1 and 5 in array. These indexes found by binary search. I called 5 parameter recursive method which takes array, ‘bound’ which we will found true position for it, index1 index2 for searching and one flag integer to recognize ‘bound’ is upper or lower bound. Operations are executed according to this flag.**

**Q3 – For third question I used additional 2 integer parameter for indexes and 1 integer parameter for sum to compare to target. Methods start from index1 = 0 and index2 = 1 and adds them to ‘sum’. If ‘sum’ is less then ‘target’ value, index2 will be increased by 1 and calling new recursive method. If ‘sum’ is greater than ‘target’ value, index1 will be increased by 1 and index2 will be increased by 2. If sum = target prints from index1 to index2 by using recursive “print” method that I implemented myself.**

**Q4 – Because only time complexity explaination is needed, running time of Q4 method is calculated in these section.**

**This given function is multiplies large two number with divide and conquer algorithm. It recursively splits the numbers, performs the required calculations, then joins the results to get the result of the multiplication. It is like merge sort algorithm.**

**For n digit integer, we have to process 3 multiplications of integers of size (n/2). Recurrence relation for this method is**

**T(n) = 3T(n/2) , for n > 1**

**T(n) = 1, for n = 1**

**T(n) = 3T(n/2)**

**T(n/2) = 3T(n/4)**

**T(n) = 3(3T(n/4)) = 32T(n/22)**

**After k iterations,**

**T(n) = 3kT(n/2k)**

**So, k = log2n ⇒ n = 2k**

**from equation T(n) = 3kT( 2k /2k)**

**T (n) = nlog23**

**>>>>>>** Θ **(n1.6) <<<<<<**

**4- Test Cases**

**Q1)**

TEST CASES

Question1("erselcelaleren","celal",1)

Question1("erselcelaleren","celal",1),"erselcelaleren", "celal")

Question1("erselersel","ersel",2),"erselersel", "ersel")

Question1("erselersel","ersel",3),"erselersel", "ersel")

Question1("ersel","erselcelaleren",1),"ersel", "erselcelaleren")

Question1("bcdbcd","bcd",1),"bcdbcd", "bcd")

Question1("bcdbcd","bcd",2),"bcdbcd", "bcd")

Question1("bcdbcd","bcd",3),"bcdbcd", "bcd")

Question1("aaaaa","a",1),"aaaaa", "a")

Question1("aaaaa","a",5),"aaaaa", "a")

Question1("abc","abcdef",1),"abc", "abcdef")

Question1("aaaaa","aaaaa",1),"aaaaa", "aaaaa")

Question1("aaaaaa","aaaaa",2),"aaaaaa", "aaaaa")

Question1("","",1),"", "")

RESULTS

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Q2)**

*TEST CASES*

*int*[] arr **=** {0,3,4,6,10,11,15,17,20,22,30}

number1 **=** **-**1;

number2 **=** 35;

-------------

number1 **=** 0;

number2 **=** 30;

-------------

number1 **=** 0;

number2 **=** 3;

-------------

number1 **=** 0;

number2 **=** 4;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu------------- RESULT

number1 **=** 3;

number2 **=** 3;

-------------

number1 **=** 4;

number2 **=** 5;

-------------

number1 **=** 16;

number2 **=** 16;

-------------

number1 **=** 15;

number2 **=** 17;

-------------

number1 **=** 15;

number2 **=** 18;

-------------

number1 **=** 15;

number2 **=** 20;

-------------

number1 **=** 30;

number2 **=** 30;

-------------

number1 **=** 0;

number2 **=** 0;

-------------

number1 **=** 1;

number2 **=** 10;

-------------

number1 **=** 100;

number2 **=** 150;

-------------

number1 **=** **-**30;

number2 **=** **-**6;

**Q3)**

*int*[] arr **=** {1,3,5,2,3,2,0,8,4,3,11,6,1,8,2,2,2,2,2,7,7}; *//son indexi saymıyor düzelt*

Question3(arr,7);

Question3(arr,10);

Question3(arr,12);

Question3(arr,14);

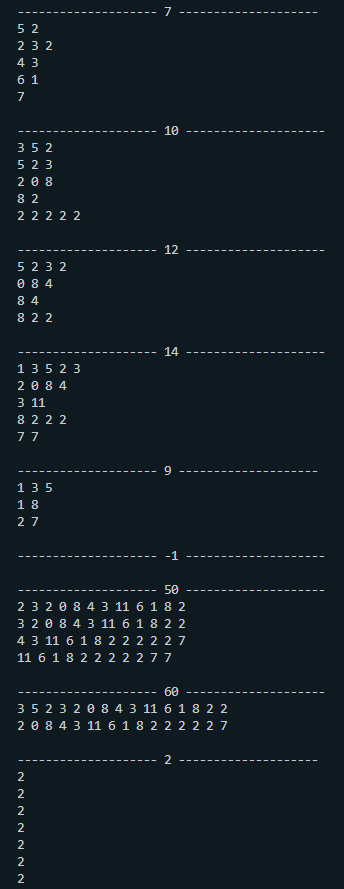
Question3(arr,9);

Question3(arr,**-**1);

Question3(arr,50);

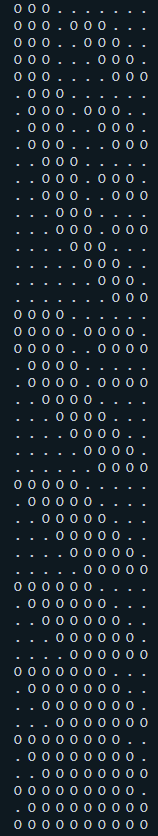
Question3(arr,60);

Question3(arr,2);

 TEST RESULTS

**Q5)**

**6 sized 7 sized 8 sized array 10 sized array**

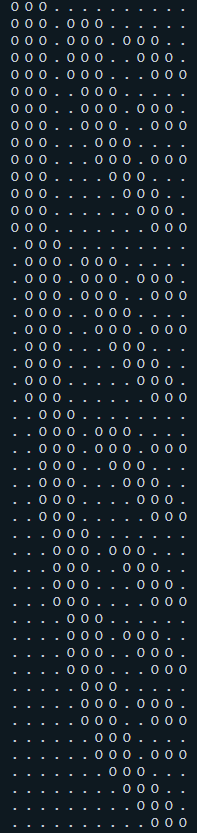
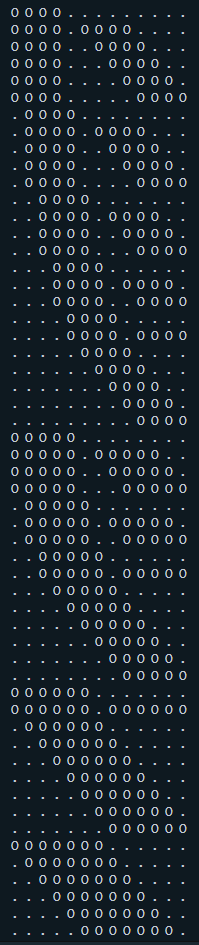
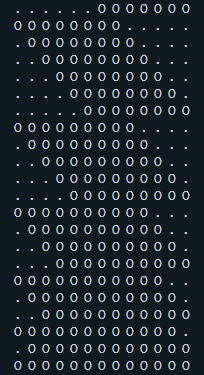
siyah, iç mekan, rende, mutfak eşyası içeren bir resim

Açıklama otomatik olarak oluşturuldukravat, gömlek, kapat içeren bir resim

Açıklama otomatik olarak oluşturuldukravat, kapat içeren bir resim

Açıklama otomatik olarak oluşturuldu

**------------------------------13 Sized Array-----------------------------**



**Complexity Analyzes**

**Q1) INDUCTION 1**

T(n) = +n)/2 + 1

T(n) = T(n-1) + n | T(0) = Θ(1) |



T(1) = T(0) + 1 = 1 🡪 T(1) = (1^2 + 1) / 2 = 1

T(2) = T(1) + 2 = 3 🡪 T(2) = (2^2 + 2) / 2 = 3

T(3) = T(2) + 3 = 6 🡪 T(3) = (3^2 + 3) / 2 = 6

T(4) = T(3) + 4 = 10 🡪 T(4) = (4^2 + 4) / 2 = 10

-------------------------------



Base Case: T(0) = (0^2 + 0) / 2 + 1 = 1 , n=0;

1 = 1 is true

Induction Hypothesis: for some arbitrary value of n

T(n) = +n)/2 +1 is true

Induction Step: T(n+1) = ((n+1)^2 + n+1) / 2 + 1 = n^2 + 3n + 3

T(n+1) = T(n+1-1) + n+1

T(n+1) = T(n) + n+1

T(n+1) = ( n^2 + n ) / 2 + n+1 + 1

T(n+1) = n^2 + 3n + 3 is true

Best case T(n) = Θ(1)

Worst case T(n) = Θ(n^2)

General complexity T(n) = O(n^2)

Q2) INDUCTION 2

Input size is ‘index2 – index1’ which is the size of array at the beginning. At each call of method this input size will be divided by 2. So it will be process like k, k/2, k/4, k/8 …

Worst case will show up if upper bound is largest element in the array and lower bound is smallest number in the array.

Worst case T(n) = Θ(logn)

Best case will be the situation which we found the exact bound in the middle of array.

Best case T(n) = Θ(1)

**General complexity T(n) = O(logn)**

T(n) = T(n/2) + Θ(1)

Base case: T(1) = 0 for n=1

log 21 = 0 is true

Induction Hypothesis:

T(n) = log2 n is true

Induction Step:

T(2n) = log 2 (2n)

T(2n) = T(2n/2)+1

T(2n) = T(n) + 1

T(2n) = log 2n + 1

log 2 (2n) = log 2n + log 22

log 2 (2n) = log 2n + 1

T(2n) = log 2n + 1 = log 2 (2n)

Q3)

Complexity Best Case T(n) = Θ(1)

Complexity Worst Case T(n) = Θ(n^2)

General complexity T(n) = O(n^2)

T(0) = Θ(1)

T(n) = T(n-1) + n

T(n) = T(n-1) + n

T(n) = T(n-2) + n -1

T(n) = T(n-3) + n - 2 + n -1

T(n) = T(n-4) + n – 3 + n – 2 + n -1

.

.

.

T(n) = T(n-k) + n.k – (k.(k-1))/2 >>> for n = k

T(n) = T(0) + n^2 - (n^2 – n)/2

T(n) = (n^2 + n)/2 + 1

**2 - Class Diagrams**

metin, siyah, skorbord, metal içeren bir resim

Açıklama otomatik olarak oluşturuldu