Hazal Gören
131044028

CSE-321 Homework - 03

1)

a. If we measure the size of an instance of the problem of computing the greatest common divisor of m and n by the size of the second parameter n, by How much can the size decrease after one iteration of Euclid's algorithm?

b.) Prove that an instace size will always decrease at least by a factor of 2 after two serccussive iterations of Euclid's algorithm. Let r=m mod n. Investigate two cases of r's value relative to n's value.

a) gcd(m,n) = gcd(n, m mod n)

→ Euclid(a, b)
if b==0
return a
else return Euclid(b, a mod b)

} algoritma

Euclid(30,21) = Euclid(21,9)
= Euclid(9,3)
= Euclid(3,0)
= 3

} 3 recursive de bulunmustur

Her recursive'de 2. argüman kesinlikle azalmaktadır.

Boyut n 1 ve n arasında azalmaktadır.

b.) gcd(m,n) = gcd(n,r) = gcd(r, n mod r)

2 recursiveden sonra 2 kat azalacağını kanıtlamanı istemiş.

$$gcd(m, n) = gcd(n, r) = gcd(r, n \bmod r)$$

boyutun en az $\frac{n}{2}$ olması gerekir.

$$\frac{n}{2} \geq n \bmod r$$

n	r	n mod r
1	0	
2	0	
3	1	
4	2	
5	0	
6	4	n − r
7	3	
8	2	
9	1	
0	0	

Bu durumda n mod r için iki farklı durum olusur.

1. durum $r \leq \frac{n}{2}$

$$n \bmod r < r \leq \frac{n}{2}$$

2. durum $\frac{n}{2} < r < n$

$$n \bmod r = n - r < \frac{n}{2}$$

2. An algorithm which ensures that each new permutation is created by exchanging only two neighboring elements is called a minimal change algorithm Design a minimal change decrease by-one algorithm for generating permutations of a given set of integers 1,2, -- ,n
Stein hous Johnson Trotter algoritması (PS'den yararlanılmıştır)

1 2 3	First the numbers 1-- n are written
1 3 2	in the increasing order and a direction
3 1 2	is assigned to each of them which
3 2 1	is initially Left. Note that < symbol
2 3 1	in front of each number below indicates
2 1 3	that the direction associated with it is
	Left.

Similary a number followed by > symbol would indicate that its direction is right.

This algorithm uses a term called mobile integer. An integer is said to be mobile, if the adjacent number on its direction is smaller than this

Algoritma

i. The algorithm works by placing the numbers 1---n in the increasing order and associating left < as the direction for each of them

ii. Find the largest mobile and swap it with the adjacent element on its direction without changing the direction of any of these two.

iii. In doing so, if the largest mobile integer has reached a spot where it's no more mobile, proceed with the next largest integer if it's mobile.

iv. After each swapping, check if there's any number, larger than the current largest mobile. If there is one or more, change the direction of all of them.

v. The algorithm terminates when there are no more mobile.

//internetten yadım aldım.

```
def permute(xs, low=0):
    if low+1 >= len(xs):
        yield xs
    else:
        for p in permute(xs,low+1)
            yield p
        for i in range(low+1, len(xs)):
            xs[low], xs[i] = xs[i], xs[low]
            for p in permute(xs, low+1);
                yield p
            xs[low], xs[i] = xs[i], xs[low]
```

3.) Outline algorithm for deleting a key from a binary search tree. Consider seperately three cases: (1) the key's node is a leaf ; (2) the key's node has one child ; (3) the key's node has two children.

a.) Would you classify this algorithm as a variable size decrease algorithm?

b.) What is the time efficiency class of your algorithm?

a) → Binary Search Tree algoritmasında eleman silmek için:

- Silinecek elemanın hiç çocuğu yok ise direk null yapılır.
- En az 1 çocuk varsa silme işlemi direk gerçekleşemez çünkü alt ağaç kaybedilmiş olur.
  Silinecek elemanın solundaki alt ağacın en büyük elemanının ya da sağındaki alt ağacın en küçük elemanının silinen elemanın yerine yerleştirilmesi gerekir.

  Bu algoritma variable-size-decrease değildir.

b) Binary Search tree $W(n) \in O(\log n)$'dir.

4.) Suppose that an array contains n numbers, each of which is -1, 0, 1. Then the array can be sorted in $O(n)$ time in the worst case. Prove or disprove that statement.

→ Counting sort algoritması -1, 0, 1 sayılarını $O(n)$ zamanda sıralar. 1

Teorik olarak n sayıda sayıyı n zamanda sıralamak mümkün değildir. Ancak bu algoritma yerden feragat ederek n zamanda sıralama yapmaktadır.

2. bir dizi tutup 1 arttırdım

-1 0 1 -1 0 1

0	1	2
2	2	2

→ sayılarını tutar

-1	0	1
2	2	2

→ 1 azalttım

→ sayılarını tutar.

-1 -1 0 0 1 1

5.) Given the array A[1...n] of sorted distinct integers, design a divide and conquer algorithm that finds an index i such that A[i]=i. Your algorithm should run in O(logn) time.

(logn) zamanda calismasi icin binary search kullandim.

A[2]=?

2<6 sol tarafa git

2<3 sol tarafa git

2=2 bulundu