Pamukkale Üniversitesi
Bilgisayar Mühendisliği
Algoritmalar Dönem Sonu Sınavı Cevap Kağıdı 10.06.2016

Öğrenci Numarası : _____ Adı Soyadı : _____

Soru	1	2	3	4	Toplam
Puan	25	25	25	25	100
Not					

1. Aşağıdaki tablonun ilk satırında verilen metin içinde "ağız" kelimesi aranmak isteniyor.
   (a) (10P) Horspool algoritması için kaydırma tablosunu oluşturun

a	ğ	ı	z	*
3	2	1	4	4

   (b) (10P) Horspool algoritmasına göre örüntüdeki karşılaştırılan karakterlerin altını çizerek arama işlemini yapın ve toplam kaç karşılaştırma yapıldığını hesaplayın.

9

   (c) (5P) Kaba kuvvet metin eşleştirme algoritması kullanılsaydı kaç adet karşılaştırma yapılırdı?

24

Table 1: Horspool

p	i	j	a	m	a	l	ı		h	a	s	t	a		y	a	ğ	ı	z		ş	o	f	ö	r	e		ç	a	b	u
a	ğ	ı	**z**																												
			a	ğ	ı	**z**																									
							a	ğ	ı	**z**																					
										a	ğ	ı	**z**																		
													a	ğ	ı	**z**															
																a	**ğ**	**ı**	**z**												

2. (25P) Tablo 2'de ağırlık ve değerleri verilen elemanlar için 0-1 sırt çantası(0-1 knapsack) problemini dinamik programlama ile çözün. Çantanın taşıyabileceği en büyük ağırlık W=10'dir.

$$F(n) = \begin{cases} \max\{F(i-1,j), v_i + F(i-1, j-w_i)\} & \text{if } j - w_i \geq 0 \\ F(i-1,j) & \text{if } j - w_i < 0 \end{cases}$$

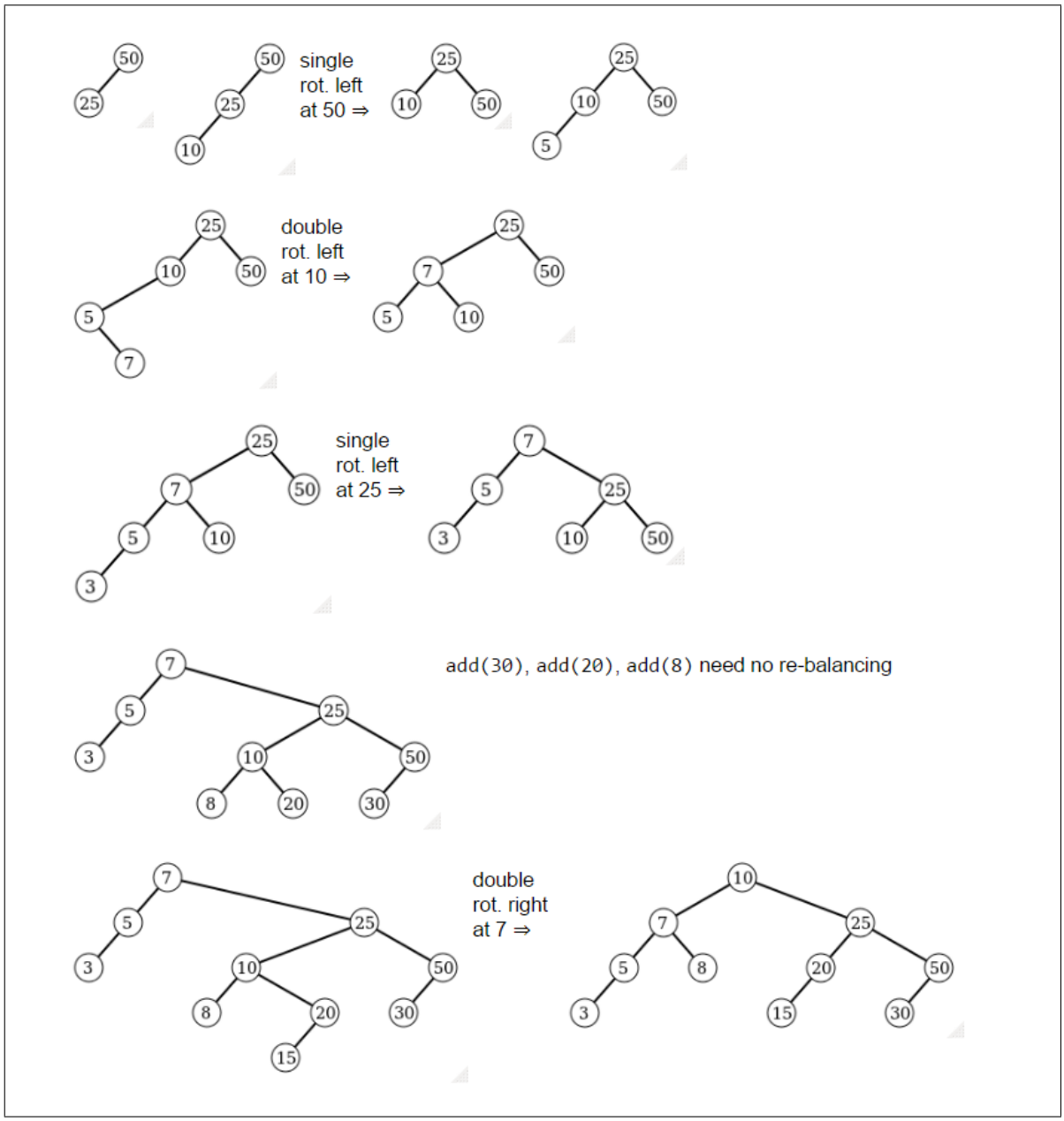
Table 2: Knapsack

Eleman	Ağırlık	Değer
1	1	10 TL
2	3	15 TL
3	2	20 TL
4	4	40 TL
5	5	20 TL
6	4	42 TL

Table 3: Knapsack çözümü

El. \Kap.	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	10	10	10	10	10	10	10	10	10	10
2	**0**	10	10	15	25	25	25	25	25	25	25
3	0	10	**20**	30	30	35	45	45	45	45	45
4	0	10	20	30	40	50	**60**	70	70	75	85
5	0	10	20	30	40	50	60	70	70	75	85
6	0	10	20	30	42	52	62	72	82	92	**102**

3. (25P) 50, 25, 10, 5, 7, 3, 30, 20, 8, 15 değerlerini sırasıyla bir AVL ağacına ekleyiniz. Döndürme işlemindekiler hariç toplamda kaç adet karşılaştırma yapılmaktadır yazınız.


single
rot. left
at 50 ⇒
double
rot. left
at 10 ⇒
single
rot. left
at 25 ⇒
add(30), add(20), add(8) need no re-balancing
double
rot. right
at 7 ⇒

4. (25P) Şekil(Tablo) 4'deki ~~ikili ağaç yapısında~~ yapıda kökten yapraklara kadar olan yollardan en büyüğü bulunmak istenmektedir. Şekil 4 için kökten itibaren 3,7,4,9 yolu izlenerek en büyük toplam değeri 23 olarak bulunabilir. Bu tipteki ağaçları temsil etmek Şekil 5'deki gibi bir matris verilmiştir. Matrisin [0,0] elemanı kökü, [1,0] elemanı kökün sol çocuğunu, [0,1] elemanı kökün sağ çocuğunu temsil etmektedir. Verilen matris için en büyük yol toplamını bulan aşağıdaki algoritmayı yazın ve bu algoritmanın analizini gerçekleştirin(temel işlem, temel işlemin tekrarlanma sayısı,verimlilik sınıfı). Algoritmada dikkat edilmesi gereken bir nokta: n+1 derinliğe sahip bir ağaçta 2^n farklı toplam değeri bulunmaktadır. Bütün toplamları ayrı ayrı hesaplamak yerine akıllı çözümler uygulanabilir.

Table 4: En büyük toplam yolu

			3			
		7		4		
	2		**4**		6	
8		5		**9**		3

Table 5: Matris

3	0	0	0
7	4	0	0
2	4	6	0
8	5	9	3

Çözüm:

Matristeki değerler en üst satırdan aşağıdaki satırlara doğru en büyük yollar seçilip toplanarak giderse en alt satırdaki en büyük değer en büyük yolu verecektir.

Algorithm 1 Ağaçta kökten yaprağa en büyük yolu bulur

```
1:  function EnBuyukYol(A[0..n-1, 0..n-1])   ▷ Giriş olarak nxn elemanlı matris veriliyor
2:      for i ← 1 to n-1 do
3:          A[i,0] ← A[i,0] + A[i-1,0]                          ▷ En soldaki eleman
4:          A[i,i-1] ← A[i,i-1] + A[i-1,i-1]                    ▷ En sağdaki eleman
5:          for j ← 1 to i-1 do
6:              A[i,j] ← A[i,j] + max{A[i-1,j], A[i-1,j+1]}
7:      enbuyuk ← A[n-1,0]
8:      for i ← 1 to n-1 do                          ▷ En alt satırdaki en büyük değeri bul
9:          if enbuyuk < A[n-1,i] then
10:             enbuyuk ← A[n-1,i]
11:     return enbuyuk                  ▷ en büyük toplamı döndürür(Örnek için 23 döner)
```

Algoritmanın temel işlemi *max* işlemidir.

$$M(n) = \sum_{i=1}^{n-1}\sum_{j=1}^{i-2} 1 = \sum_{i=1}^{n-1}(i-2) = \sum_{i=1}^{n-1} i - 2\sum_{i=1}^{n-1} 1 = \frac{n(n-1)}{2} - 2n + 2 = \frac{n^2-5n+4}{2}$$