Hazal Gönen
131044028

CSE 321 - Introduction to Algorithm Design
Homework 04

Deadline: 23:55 December 12th, 2016
PS: Upload your homework to Moodle website. Do <u>not</u> bring papers to the room of the TA.

1. Design a divide and conquer algorithm for polynomial evaluation. (For convenience assume that the polynomial is of degree (n-1) and n=2m where m is a positive integer). Calculate the number of additions and multiplications required by your algorithm.

//internetten yardım aldım.

fft($n, a_0, a_1 \ldots a_{n-1}$){

if(n= =1)

return a_0

$(e_0, e_1, e_2 \ldots e_{n/2-1}) \leftarrow fft(n/2, a_0, a_2, a_4 \ldots a_{n-2})$

$(d_0, d_1, \ldots d_{n/2-1}) \leftarrow fft(n/2, a_1, a_3, \ldots a_{n-1})$

for k=0 to n/2-1 {

$w^k \leftarrow e^{2\pi i k/2}$

$y_k \leftarrow e_k + w^k d_k$

$y_{k+n/2} \leftarrow e_k - w^k d_k$

}

return $(y_0, y_1 \ldots y_{n-1})$

}

Hazal Gönen
131044028

2. Given an 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(log n) time.

Algoritmanın O(logn) de çalışması için ikili arama yöntemi kullanılmalıdır. Önce ortadaki elemana bakılmalıdır. A[i] = i ise true return edilir.

A[i] > i ise sol taraftan recursive yapılır.
A[i] < i ise sağ tarafta recursive yapılır.

```
Find (A[1 ... n], first, last)
    int mid = (first + last) / 2
    if (A[mid] == mid)
        return true;
    if (first >= last)
        return false;
    else if (A[mid] > mid)
        return Find (A[1 .. n], first, mid-1)
    else
        return Find (A[1 .. n], mid+1, last)
```

4. Write a pseudocode for a divide-and-conquer algorithm for finding values of both the largest and smallest elements in an array of n numbers.

```
Find( A[s.. f], max, min){

  if(f == s){   // 1 elemon varsa
      max = A[f]
      min = A[f]
  }
  else if(f-s == 1)   // 2 elemon varsa

      if( A[f] < A[s]){
            min = A[f];
            max = A[s];
      }
      else {
            min = A[s];
            max = A[f];
      }
  }
  else{
      Find( A[s ..... f/2], max, min)
      Find( A[f/2+1 .... f], max1, min1)
      if(min1 < min){
            min = min1;
      }
      if(max1 > max){
            max = max1;
      }
  }
}
```

5. You are given a chocolate bar puzzle given as an n-by-m chocolate bar. You need to break it into nm 1-by-1 pieces. You can break a bar only in a straight line, and only one bar can be broken at a time. Design an algorithm that solves the problem with the minimum number of bar breaks. What is this minimum number? Justify your answer by using properties of a binary tree. Breaking the chocolate bar can be represented by a binary tree.

- $m \times n$ olan parçayı yatay veya dikey ortadan ikiye bölmeliyiz
- 1 birimlik parça kalana kadar ikiye bölme işlemi devam etmelidir.
- 1 parça zaten olduğu için $m \times n - 1$ adıma ihtiyaç vardır.

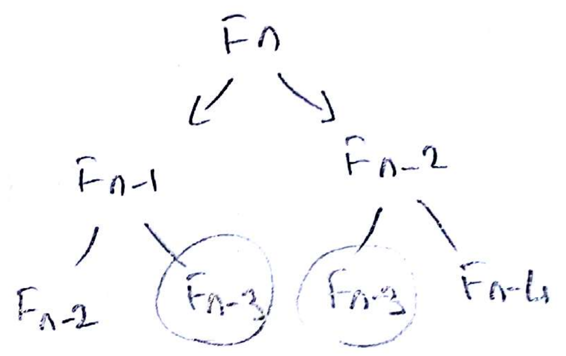
6. Questions are given below. Please answer them in detail.

a. What does dynamic programming have in common with divide-and conquer?

b. What is the principal difference between the two techniques?

a.) Divide and conquer ve dynamic programming algoritmaların ikisi de problemi daha küçük problemlere bölerek çözer. Her alt problemi tekrar ele alıp çözümleri birleştirerek çalışır.

b.) Dinamik programlamada sub problemlerin sonuçları saklanır ve böylece bir defa hesaplanmaları yeterlidir. Ancak divide & concuer algoritmasında her alt problem önceden çözülse bile tekrar çözülmek zorundadır.



Örnekte divide & conquer algoritması her seferinde F_{n-3}'ü hesaplamaktadır. Dynamic programlama ise bir defa hesaplar ve sonra hesapladığı değeri kaydedip onu kullanır.

Bu şekilde çözüm bulmak hızlanır.

7. *World Series odds:* Consider two teams, *A* and *B*, playing a series of games until one of the teams wins *n* games. Assume that the probability of *A* winning a game is the same for each game and equal to *p*, and the probability of *A* losing a game is $q = 1 - p$. (Hence, there are no ties.) Let *P(i, j)* be the probability of *A* winning the series if *A* needs *i* more games to win the series and *B* needs *j* more games to win the series.

a. Set up a recurrence relation for *P(i, j)* that can be used by a dynamic programming algorithm.

b. Find the probability of team *A* winning a seven-game series if the probability of it winning a game is 0.4.

c. Write a pseudocode of the dynamic programming algorithm for solving this problem and determine its time and space efficiencies.

a) A'nın kazanması için i kadar fazla seri kazanması gerekir.
B'nin kazanması için j kadar fazla seri kazanması gerekir.

$p \Rightarrow$ A'nın kazanma olasılığı $j, i-1$

$q = 1-p \Rightarrow$ B'nin kazanma olasılığı $i, j-1$

$$P(i,j) = p \cdot P(i-1, j) + q \cdot P(i, j-1)$$

$$P(0, j) = 1$$

$$P(i, 0) = 0$$

b.)

i \ j	0	1	2	3	4
0		1	1	1	1
1	0	0,4	0,64	0,78	0.87
2	0	0,16	0,35	0,52	0,66
3	0	0,08	0,18	0,32	0,46
4	0	0,03	0,09	0,18	0,29

Tabloyu doldururken $P(i,j) = p \cdot P(i-1, j) + q \cdot P(i, j-1)$ denklemi kullanılır.

$P(0,j) = 1$ ile doldurulur $\quad p = 0,4 \quad q = 0,6$

$P(i,0) = 0$ ile doldurulur.

$P(1,1) = p \cdot \underbrace{P(0,1)}_{1} + q \cdot \underbrace{P(1,0)}_{0} = 0.4 \cdot 1 = 0,4$

$P(1,2) = p \cdot \underbrace{P(0,2)}_{1} + q \cdot \underbrace{P(1,1)}_{1} = 0,4 \cdot 1 + 0,6 \cdot 0.4 = 0,64$

$P(1,3) = p \cdot P(0,3) + q \cdot P(1,2) = 0,4 \cdot 1 + 0,6 \cdot 0.64 = 0,784$

$P(1,4) = p \cdot P(0,4) + q \cdot P(1,3) = 0,4 \cdot 1 + 0.6 \cdot 0,78 = 0,868$

$P(2,1) = p \cdot P(1,1) + q \cdot P(2,0) = 0{,}4 \times 0.4 + 0.6 \cdot 0 = 0{,}16$

$P(2,2) = p \cdot P(1,2) + q \cdot P(2,1) = 0{,}4 \times 0{,}64 + 0.6 \cdot 0.16 = 0{,}352 \sim 0{,}35$

$P(2,3) = p \cdot P(1,3) + q \cdot P(2,2) = 0.4 \times 0{,}78 + 0{,}6 \cdot 0{,}35 = 0{,}522 \sim 0{,}52$

$P(2,4) = p \cdot P(1,4) + q \cdot P(2,3) = 0{,}4 \times 0{,}87 + 0{,}6 \cdot 0{,}52 = 0{,}66$

$P(3,1) = p \cdot P(2,1) + q \cdot P(3,0) = 0{,}4 \times 0{,}16 + 0{,}6 \cdot 0 = 0{,}064 \sim 0{,}06$

$P(3,2) = p \cdot P(2,2) + q \cdot P(3,1) = 0.4 \times 0{,}35 + 0{,}6 \cdot 0{,}06 = 0{,}176 \sim 0{,}18$

$P(3,3) = p \cdot P(2,3) + q \cdot P(3,2) = 0.4 \times 0{,}52 + 0{,}6 \cdot 0{,}18 = 0{,}316 \sim 0{,}32$

$P(3,4) = p \cdot P(2,4) + q \cdot P(3,3) = 0.4 \times 0{,}66 + 0{,}6 \cdot 0{,}32 = 0{,}456 \sim 0{,}46$

$P(4,1) = p \cdot P(3,1) + q \cdot P(4,0) = 0.4 \times 0{,}06 + 0{,}6 \cdot 0 = 0{,}024 \sim 0{,}03$

$P(4,2) = p \cdot P(3,2) + q \cdot P(4,1) = 0.4 \times 0{,}18 + 0{,}6 \; 0{,}03 = 0{,}09$

$P(4,3) = p \cdot P(3,3) + q \cdot P(4,2) = 0.4 \times 0{,}32 + 0{,}6 \cdot 0{,}09 = 0{,}182 \sim 0{,}18$

$P(4,4) = p \cdot P(3,4) + q \cdot P(4,3) = 0.4 \times 0{,}46 + 0{,}6 \cdot 0{,}18 = 0{,}292 \sim 0{,}29$

$P[4,4] = 0{,}29$ //

Scanned by CamScanner

c.) Algorithm WorldSeriesOdds(. n , p)

q ← 1-p

for j ← 1 to n do

P[0, j] ← 1.0

for i ← 1 to n do

P[i, 0] ← 0.0

for j ← 1 to n do

n+1 n+1 P[i, j] ← p * P[i-1-j] + q * P[i, j-1]

return P(n, n)

time ve space efficiency ⇒ $(n+1)\cdot(n+1) = n^2$ $\Theta(n^2)$

tablonun her bölümü $\Theta(1)$ zamanda doldurulur.

aa.py - C:\Users\asus\Desktop\aa.py (3.5.2)

File Edit Format Run Options Window Help

```python
# Bu kod internetten alinmistir.
import sys

# Matrix Ai has dimension p[i-1] x p[i] for i = 1..n
def MatrixChainOrder(p, n):
        # For simplicity of the program, one extra row and one
        # extra column are allocated in m[][]. 0th row and 0th
        # column of m[][] are not used
        m = [[0 for x in range(n)] for x in range(n)]

        # m[i,j] = Minimum number of scalar multiplications needed
        # to compute the matrix A[i]A[i+1]...A[j] = A[i..j] where
        # dimension of A[i] is p[i-1] x p[i]

        # cost is zero when multiplying one matrix.
        for i in range(1, n):
                m[i][i] = 0

        # L is chain length.
        for L in range(2, n):
                for i in range(1, n-L+1):
                        j = i+L-1
                        m[i][j] = sys.maxint
                        for k in range(i, j):

                                # q = cost/scalar multiplications
                                q = m[i][k] + m[k+1][j] + p[i-1]*p[k]*p[j]
                                if q < m[i][j]:
                                        m[i][j] = q

        return m[1][n-1]

# Driver program to test above function
arr = [1, 2, 3 ,4]
size = len(arr)

print("Minimum number of multiplications is " +
        str(MatrixChainOrder(arr, size)))
# This Code is contributed by Bhavya Jain
```