MID II Examination

CS204: ALGORITHMS

Solutions

Full Marks: 60 Time: 2 hour

02-03-II Date: 17-2-03

1. Solve the following recurrence equations exactly. Express your answer using theta notation. [4+6]

(ii)

$$t_n = \begin{cases} n & \text{if } n = 0 \text{ or } n = 1 \\ 5t_{n-1} - 6t_{n-2} & \text{otherwise} \end{cases}$$

(ii)

$$t_n = \begin{cases} 1 & \text{if } n = 0 \\ 4t_{n-1} - 2^n & \text{otherwise} \end{cases}$$

Answer: (i) The characteristic equation for this recurrence is $x^2 - 5x + 6x = 0$. The roots are $r_1 = 3$ and $r_2 = 2$. Thus the general solution to the recurrence is $t_n = c_1 3^n + c_2 2^n$. Using boundary conditions we get $c_1 = -c_2 = 1$. Therefore, the solution is $t_n = 3^n - 2^n$. Thus the solution is $\Theta(3^n)$.

(ii) First rewrite the equation as

$$t_n - 4t_{n-1} = -2^n.$$

Replace n by $n - 1$ in above and multiply the resulting equation by 2. This gives

$$2t_{n-1} - 8t_{n-2} = -2^n$$

Now subtract the second from first

$$t_n - 6t_{n-1} + 8t_{n-2} = 0$$

The characteristic equation is thus $x^2 - 6x + 8 = 0$ with roots as $r_1 = 4$ and $r_2 = 2$. This implies that the general solution is

$$t_n = c_1 4^n + c_2 2^n$$

Now to find c_1 and c_2 we first notice that

$$\begin{aligned} -2^n &= t_n - 4t_{n-1} \\ &= c_1 4^n + c_2 2^n - 4(c_1 4^{n-1} + c_2 2^{n-1}) \\ &= -c_2 2^n \end{aligned}$$

Thus $c_2 = 1$ regardless of the boundary conditions. So, $t_n = c_1 4^n + 2^n$. Applying intial condition for $n = 0$, we find $1 = c_1 4^0 + 2^0$, or $1 = c_1 + 1$, giving $c_1 = 0$. Hence the solution to above recurrence is $t_n = 2^n$ or $\Theta(2^n)$.

2. Give the direct proof for case 2 of the master theorem. [6]

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ aT(n/b) + f(n) & \text{if } n = b^i \end{cases}$$

Answer: Let $b > 1$ and n be an exact power of b, i.e., $n = b^k$ for $k > 0$. So

$$T(n) = \begin{cases} \Theta(1) & \text{, if } n = 1 \\ aT(n/b) + f(n) & \text{, if } n = b^i \end{cases}$$

where i is a positive integer.

Now apply iterative substitution

$$T(n) = f(n) + af(n/b) + \ldots + a^{\log_b n} T(1)$$

Since $a^{\log_b n} = n^{\log_b a}$ $a^{\log_b n} T(1) = n^{\log_b a}$,

$$T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$$

In case 2,

$$f(n) = \Theta(n^{\log_b a}) \Rightarrow f(n/b^j) = \Theta((n/b)^{\log_b a})$$

Thus

$$\begin{aligned} \sum_{0}^{\log_b n - 1} a^j \cdot f(n/b^j) &= \sum_{0}^{\log_b n - 1} a^j \cdot (n/b^j)^{\log_b a} \\ &= n^{\log_b a} \sum_{0}^{\log_b n - 1} (a/(b^{\log_b a}))^j \\ &\leq n^{\log_b a} \log_b n \end{aligned}$$

3. Vashmashur can either walk in steps of 1 meter or in steps of 2 meters. Design a recursive algorithm to calculate the number of ways Vashmashur can walk a distance of n meters to catch Shiva. You must first provide a recursive formulation of $v(n)$ which may be taken as the number of ways Vashmashur can cover the distance of n meters, clearly specifying the base cases of the recurrence. Do you find this recurrence familiar? If so, what is it? [9]

Answer: The base cases of recursion are: $v(1) = 1$, $v(2) = 2$. Since the walk must begin by either an 1 meter walk or a 2 meters walk, we can account for all the walks by following recurrence relation

$$v(n) = v(n-1) + v(n-2)$$

Combining with base case, the recurrence that defines Vashmashur's walk is

$$v(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ v(n-1) + v(n-2) & \text{if } n \geq 3 \end{cases}$$

This recurrence is identical to Fibonacci series. We have

$$v(1) = 1, v(2) = 2, v(3) = 3, v(4) = 5, v(5) = 8, v(6) = 13, \ldots.$$

Recursive algorithm which computes Vashmashur's walk is:

```
VashmashurWalk(n) {
    if ((n = 1) || (n = 2))
        return n;
    return (( Vashmashur(n − 1) + (Vashmashur(n − 2))
}
```

4. Study Euclid's gcd algorithm shown below

```
gcdAlgorithm(int m, int n) {
    int r;
    while (n>0) {
        r = m mod n;
        m = n;
        n = r;
    }
    return m;
}
```

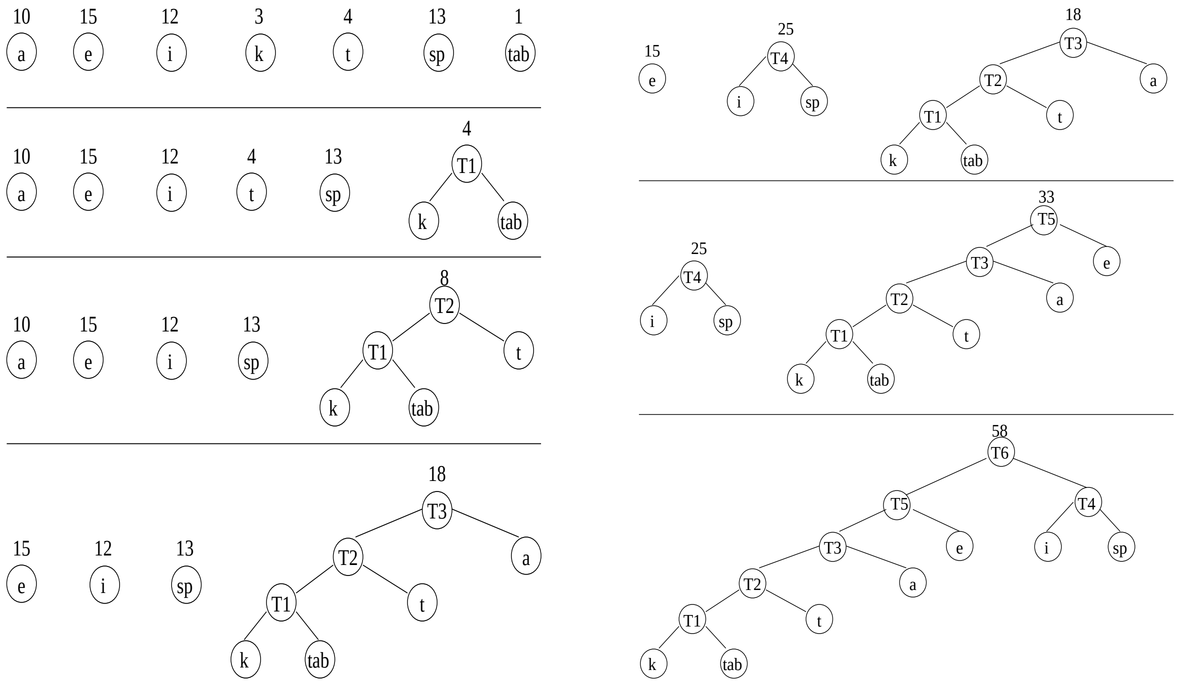
Prove that the running time of this algorithms is $O(\log n)$. In particular show that the number of iteration is at most $2 \log n$. [9]

Answer: In the above algorithm, last non-zero remainder gives the gcd. After each iteration `n` is set to `r` computed during that iteration. Algorithm terminates when `n` becomes 0. Therefore, the running time of `gcd` algorithm will be determined by the length of remainder sequence. We first prove that if `m>n` then `r < m/2`. Since `r = m` **mod** `n`, if `n < m/2` then we have nothing to prove, as the remainder of a division cannot be greater than the divisor. But if `n > m/2` then `n` goes once into `m` and the remainder resulting from this division is `m - n < m/2`. So, in both cases `r < m/2`. Before the next iteration starts `m` set to `n` and `n` is set to `r`. So, the value of `m` is definitely reduced to at most half its value after 2 iterations. Hence the result.

5. Find the optimal Huffman code for the following set of characters with the respective frequencies tabulated below. The solution must also include intermediate steps of contruction of optimal code. [7]

Character	Frequency
a	10
e	15
i	12
k	3
t	4
<tab>	1
<sp>	13

Answer: The solution is as illustrated by the figure below



6. Prove that all maximal independent sets in a matroid have same cardinality. [4]

   Answer: Assume that A and B are two maximal independent sets. Let $|A| < |B|$. Then it implies that there is some element $x \in B$ such that $x \notin A$. Then by exchange property A is extendable to $A \cup \{x\}$. It contradicts the maximality of A.

7. (a) Prove that every triangulation of an n-vertex convex polygon has $n-3$ chords and divides the polygon into $n-2$ triangles. [7]

   Answer: Proof is by induction on the number of vertices in the given polygon.

   Base case. When $n = 3$, the polygon is a triangle. it has no chord. So number of chords is $n - 3 = 3 - 3 = 0$, and number of triangles is $n - 2 = 3 - 2 = 1$.

   Induction hypothesis. Let the result hold for a convex polygon with $n = k$ vertices.

Induction step. Consider a convex polygon P with $n = k + 1$ vertices. Draw a chord joining any two non-adjacent vertices. The chord divides the polygon into two convex polygons P_1 and P_2. Let there be k' vertices in P_1. So the number of vertices in $P - 2$ is $(k + 1) - k' + 2$. By induction hypothesis P_1 has $k' - 3$ chords and $k' - 2$ triangles. Similarly by induction hypothesis P_2 has $(k+1)-k'-1$ chords and $(k+1)-k'$ triangles. So the number of chords in P is

$$(k + 1) - k' - 1 + k' - 2 = (k + 1) - 3, \text{ and } (k + 1) - k' + k' - 3 = (k + 1) - 3 \text{ triangles.}$$

Hence, the result follows by induction.

(b) Show how to construct LCS from the following c-table. Formulate the process you employed as a simple recursive algorithm to construct the LCS of any sequence $X =< x_1, \ldots, x_m >$ and another sequence $Y =< y_1, \ldots, y_n >$ in time $\mathrm{O}(m + n)$. [8]

	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

Answer: Start constructing LCS from $c[7, 6]$. Since $x[7] \neq y[6]$ and $c[7, 6] \geq c[7, 5]$ call construction of LCS recursively from $c[6, 5]$. Notice that $x[6] = y[5]$ and $c[6, 5] = c[5, 4] + 1$ thus recursively call the LCS construction from $c[5, 4]$, following which print the value in $x[6]$. Proceeding like this trace the path as shown, in the table below, and construct LCS $= < B, C, B, A >$.

i \ j		0	1	2	3	4	5	6
		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

The algorithm suggested above is as follows:

```
printLCS(c, X, Y, i, j) {
    if ((i==0) || (j==0))
        return;
    else if ((c[i,j] == c[i-1,j-1]+1) && (X[i]==Y[j])) {
            printLCS(c, X, Y, i-1, j-1);
            print x[i];
        }
        else if (c[i-1,j] ≥ c[i, j-1])
                printLCS(c, X, Y, i-1, j);
            else
                printLCS(c, X, Y, i, j-1);
}
```