CSci 242: Algorithms and Data Structures

Instructor: Dr. M. E. Kim Date: November 15th, 2018

Exam 2: Solution

Q1. [30] **Short Answer**

Explain your answer clearly.

1) [5] Give the name of one of the sorting algorithms whose running time is **O(*n*)**.

   Bucket sort, Radix sort,

2) [5] Give the names of sorting algorithm which was designed based on ***Divide & Conquer paradigm.***

   Merge sort, Quick sort

3) [10] Give the ***recurrence equation*** for the running time of Quick Sort algorithm both (A) in the ***worst*** case and (B) in the ***best*** case. Then, (C) give their solutions of the running time in Big-Oh (**O**) notation.

   Worst case: $T(n) = \begin{cases} O(1) & n = 1 \\ T(n-1) + O(n) & n > 1 \end{cases}$ $\Rightarrow$ $T(n) = O(n^2)$

   Best case: $T(n) = \begin{cases} O(1) & n = 1 \\ 2T\left(\frac{n}{2}\right) + O(n) & n > 1 \end{cases}$ $\Rightarrow$ $T(n) = O(n \log n)$

4) [5] Describe the algorithm design paradigm of ***Divide and Conquer***.

   A general algorithm design paradigm that designs a recursive algorithm in 3 steps:

   1. Divide: divide the problem of the input S into a number of subproblems with disjoint subsets of inputs S_1, S_2, ...
   2. Conquer: recursively solve the subproblems with the data S_1, S_2, ...
   3. Combine: combine the solutions for S_1, S_2 into a solution for S.

5) [5] In the data encoding/decoding of the characters, what is/are the least requirement to achieve the ***optimal codes with no ambiguity***?

   A variable length code and a prefix code.

Q2. [20] **Job Scheduling Problem**

Suppose a hair stylist has several customers waiting for different treatments. The treatments don't all take the same amount of time, but the stylist knows how long each takes. A reasonable goal would be to schedule the customers in such a way as to minimize the total time they spend both waiting and being served, which is called the time in the system.
This is called *a problem of minimizing the total time in the system.:* total time = waiting time + service time.

Five customers and their service times are given below:

Customer	Service Time (min.)
1	40
2	20
3	80
4	50
5	60

1) [10] Write a recursive **or** an iterative greedy algorithm, **Schedule(Customer, ?)**, that both decide ***the optimal sequence*** of customers to minimize the total time spent in the system and also computes ***the total system time*** in the system.
An input is given in the array named Customer[1 .. N] in which Customer[*i*] stores a service time of customer-*i*.
You can define more arguments '?' of the algorithm if they're needed.

```
        Sort an array of Jobs in ascending order of service-time;
// initial i = 1, Total =Wait = 0

Algorithm RecSchedule(i, , Wait, Total) {
        Wait = Wait + Jobs[i].service-time
        Total = Total + Wait
        If i < N then RecSchedule(i+1, Total)   // N is the total number of jobs in the array.
        return (Jobs, Total)
```

See the similar algorithm that choose the items to maximize the benefit.in the slide #9 - #10.

```
Algorithm itSchedule(Jobs)

        Sort an array of Jobs in ascending order of service-time;

        waiting = 0, Total = 0, i = 1
        while (i ≤ N)  { //  N is the total number of jobs
                waiting = waiting + Jobs[i].service  // the updated cumulative waiting time
                Total = Total + waiting;
                i = i + 1                    }
        return Jobs,  Total
```

Algorithm itSchedule(Jobs)

Sort the Jobs in the array in ascending order of their service time.

i = 1; Waiting = 0; Total = 0;

While (i ≤ N) { // N is the number of jobs, i.e. the size of array

Select a job *i*.

Total = Total + (Waiting + service[i]) ; // choose the shortest service time job.

Waiting = Waiting + service[i]; // the updated cumulative waiting time

i = i+1 }

return Jobs, Total

2) [10] (A) What is the ***optimal solution*** of the above problem, i.e. the optimal sequence of customers? and (B) What is its ***minimum total system time***?

(A). 2, 1, 4, 5, 3

(B). 20/2 + (20+40)/1 + (20+40+50)/4 + (20+40+50+60)/5 + (20+40+50+60+80)/3 = 20+60+110+170+250 = 610

Q3. [20] **Divide & Conquer**

1) [10] Write a in-place recursive algorithm, named **Minimum(A, *a, b*)**, based on the Divide & Conquer paradigm to find the minimum element in the array A[*a* .. *b*]. Suppose the number of elements stored in A is $n = 2^k$, i.e. it's always dividable by 2.

(A)

Algorithm Minimum(A, *a, b*)

If *a* = *b* then return A[*a*]

p ← ⌊(*a*+*b*)/2⌋

m1 ← Minimum (A, *a, p*)

m2 ← Minimum (A, *p*+1, *b*)

return min (m1, m2)

(B) If the idea of QuickSort is used,

Algorithm Minimum(A, *a, b*)

If *a* ≥ *b* then return A[*b*]

p ← InPlacePartition(A, a, b)

m ←Minimum (A, *a, p*)

return m

Use InPlacePartition algorithm in the textbook. (slide #44-Chap. 8)

2) [5] Write a ***recurrence equation*** for the running time of the algorithm in 1).

(A) $T(n) = \begin{cases} O(1) & n = 1 \\ 2T\left(\frac{n}{2}\right) + O(1) & n > 1 \end{cases}$

(B) $T(n) = \begin{cases} O(1) & n = 1 \\ T\left(\frac{n}{2}\right) + O(n) & n > 1 \end{cases}$ if a good pivot

3) [5] By Master's Theorem, give the ***solution*** of the recurrence in Big-Theta (Θ)

(A) Case 1: $f(n) = O(1) = O(n^0)$

$a = b = 2$, so $\log_2 2 = 1$

So, $f(n) = n^0 = O(n^{1-1})$. Thus, $T(n) = \Theta(n^{\log_2 2}) = \Theta(n^1) = \Theta(n)$.

(B) Case 3: $f(n) = O(n)$

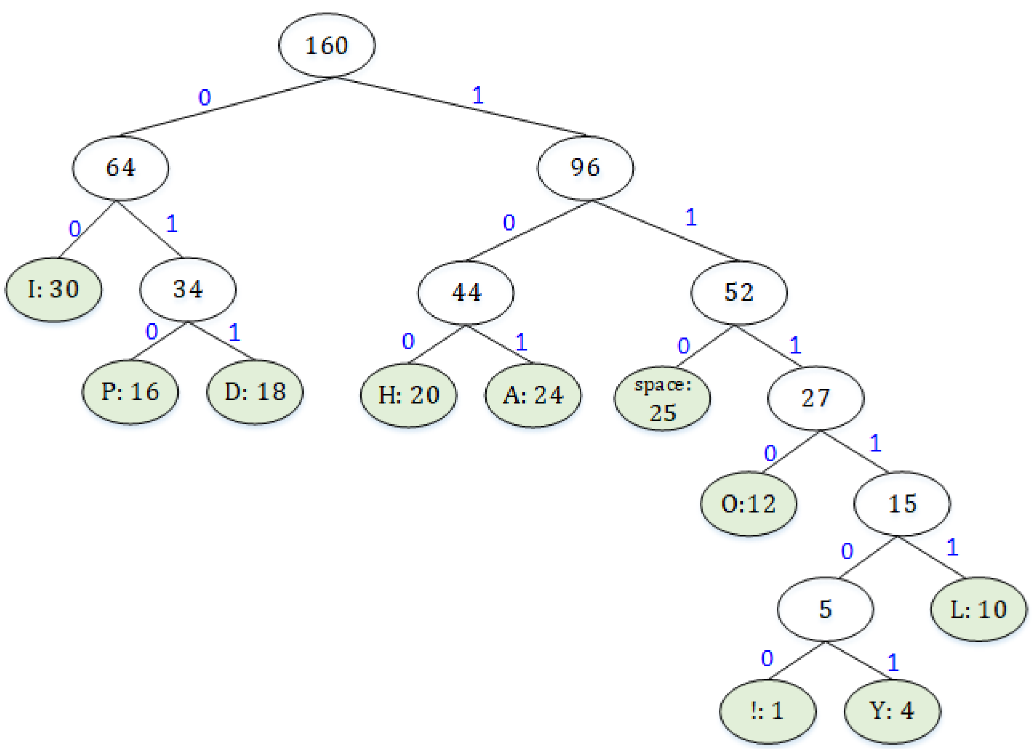
$a = 1$, $b = 2$, so $\log_2 1 = 0$

So, $f(n) = n = O(n^{0+1})$. Thus, $T(n) = \Theta(f(n)) = \Theta(n)$.

Q4. [20] **Huffman Codes**

1) [10] The text file contains only the characters shown in the table with the given frequency. Generate the ***optimal Huffman code*** for each character in the table.

character	A	I	O	Y	D	H	L	P	!	space
frequency	24	30	12	4	18	20	10	16	1	25
code	101	00	1110	111101	011	100	11111	010	111100	110



2) [5] What is the ***total number of bits*** required to encode the text file using your Huffman codes?

3*24/A + 2*30/I + 4*12/O + 6*4/Y + 3*18/D + 3*20/H + 5*10/L + 3*16/P + 6*1/! + 3*25/space

= 497 bits.

3) [5] Decode the following codes into a text.

100 101 010 010 111101 110 100 1110 11111 00 011 101 111101 111100

H A P P Y _ H O L I D A Y !

Q5. [10] **Recurrence**

For a given recurrence equation below,

$$T(n) = \begin{cases} 1 & n < 3 \\ 2T(n/3) + n & n \geq 3 \end{cases}$$

1) [10] Solve it by ***Master's Theorem***. Clearly state the case to which it belongs, the rationale of the solution and its solution in the big-Theta(Θ) notation.

a = 2, b = 3, f(n) = n^1;

$\log_b a = \log_3 2 < 1 = \log_3 3$.

Since $f(n) = n^1 = \Omega(n^{\log_3 2+\varepsilon})$ and $2 \cdot f(n/3) = 2/3 \cdot n \leq 2/3 \cdot n$ for $2/3 < 1$, this is the case 3.

Thus, T(n) = Θ(f(n)), i.e. T(*n*) = Θ(*n*).

2) [10, optional] Solve it **either** (2A) by Recursion Tree method **or** (2B) by Iterative Substitution method.

(2A) In Recursion Tree, draw it by specifying:

(a) the height of tree, (b) the number of leaves, level, (c) the number of nodes per level, (d) the size of input per level, (e) a per-level time, (f) the total time **and** (g) its asymptotic tight bound (i.e. the asymptotic tight bound of the solution in the big-Theta),

(2B) In Iterative Substitution:

The proper number of iterative steps and the final step where the size of input is 1, the computation of some parameters.

$T(n) = 2T(n/3) + n = 2\,(2\,T(n/3^2) + n/3\,) + n$

$= 2^2\,T(n/3^2) + 2n/3 + n = 2^2\,(2\,T(n/3^3) + n/3^2\,) + \frac{2}{3}n + n$

$= 2^3\,T(n/3^3) + (\frac{2}{3})^2 n + \frac{2}{3}n + n = 2^3\,(2\,T(n/3^4) + n/3^3) + (\frac{2}{3})^2 n + \frac{2}{3}n + n$

$= 2^4\,T(n/3^4) + (\frac{2}{3})^3 n + (\frac{2}{3})^2 n + \frac{2}{3}n + n = \ldots\ldots$

$= \ldots.$

$= 2^h\,T(n/3^h) + \sum_{k=0}^{h-1}(\frac{2}{3})^k n$

$= 2^{\log_3 n}\, T(1) + n\, \frac{1-(\frac{2}{3})^{\log_3 n}}{1-\frac{2}{3}}$ until $\frac{n}{3^h} = 1 \Leftrightarrow h = \log_3 n$

$= n^{\log_3 2} \cdot 1 + 3n(1 - n^{\log_3 \frac{2}{3}})$

since $\log_3 2 < 1$ and $\log_3 \frac{2}{3} = \log_3 2 - \log_3 3 = \log_3 2 - 1 < 0$

$= n^{\log_3 2} + 3n - \frac{3n}{n} n^{\log_3 2} = 3n - 2n^{\log_3 2}$

$= \Theta\,(n)$