Exercise 7 (for grade) ~ Wednesday, November 2, 2022 ~ CPSC 535.01 Fall 2022

Write one submission for your entire group, and write all group members' names on that submission. Turn in your submission before the end of class. The X symbol marks where you should write answers.

Recall that our recommended problem-solving process is:

1. **Understand** the problem definition. What is the input? What is the output?
2. **Baseline** algorithm for comparison
3. **Goal** setting: improve on the baseline how?
4. **Design** a more sophisticated algorithm
5. **Inspiration** (if necessary) from patterns, bottleneck in the baseline algorithm, other algorithms
6. **Analyze** your solution; goal met? Trade-offs?

Follow this process for each of the following computational problems. For each problem, your submission should include:

1. State are the input variables and what are the output variables
2. Pseudocode for your baseline algorithm, that needs to include the data type and an explanation for any variable other than input and output variables
1. The -notation time complexity of your baseline algorithm, with justification.

and if you manage to create an improved algorithm:

3. Answer the question: how is your improved algorithm different from your baseline; what did you change to make it faster?
4. Pseudocode for your improved algorithm, that needs to include the data type and an explanation for any variable other than input and output variables
1. The -notation time complexity of your improved algorithm, with justification.

Today's problems are:

1.

Compatible intervals: Given 10 open intervals (a1, b1), (a2, b2), . . . , (a10, b10) on the real line, each representing start and end times of some activity requiring the same resource, the task is to find the largest number of these intervals so that no two of them overlap.

a[ ]	1	6	3	8	11	8	4	5	11	14
b[ ]	2	7	9	10	12	12	7	8	13	15

Show the schedule for each of the three greedy algorithms based on:

a. earliest start first.

b. shortest duration first.

c. earliest finish first.

For each of the three algorithms, state whether the algorithm yields an optimal solution or not.

2.

Given the following set of frequencies:

a:13 b:21 c:35 d:38 e:74 f:112 g:189

a) Write the fixed-length code for each character

b) Write the Huffman code for each character. Show your work on computing Huffman code.

3.

If the symbols of the alphabet have their frequencies in increasing order then prove that the algorithm shown in class that builds Huffman tree has linear time complexity. An example is Exercise 2, above.

Names

Write the names of all group members below.

Kevin Jasani, John Tu, Hetal Patel

Exercise 1: Solve and provide answer

Start Time	a[]	1	6	3	8	11	8	4	5	11	14
End Time	b[]	2	7	9	10	12	12	7	8	13	15
Activity	Index	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

(a) Earliest Start time :-

Start time	a[]	1	3	4	5	6	8	8	11	11	14
End time	b[]	2	9	7	8	7	10	12	12	13	15
Activity	Index	(1)	(3)	(7)	(8)	(2)	(4)	(6)	(5)	(9)	(10)

Selection of activities = { 1, 3, 5, 10 }

Activity 7 overlaps with 3 so cannot be added
Activity 8 overlaps with 3 so cannot be added
Activity 2 overlaps with 3 so cannot be added
Activity 4 overlaps with 3 so cannot be added
Activity 6 overlaps with 3 so cannot be added
Activity 9 overlaps with 5 so cannot be added

The activities yield are less

∴ Not an optimal solution

(b) Shortest duration first :-

Start time	a[]	1	6	11	14	8	11	4	5	8	3
End time	b[]	2	7	12	15	10	13	7	8	12	9
Activity	Index	(1)	(2)	(5)	(10)	(4)	(9)	(7)	(8)	(6)	(3)

Selection of Activities = { 1, 2, 5, 10, 4 }

Activity 9 overlaps with 5 so cannot be added
Activity 8 overlaps with 7 so cannot be added
Activity 6 overlaps with 4, 5 so cannot be added
Activity 3 overlaps with 7 so cannot be added
Activity 7 overlaps with 2 so cannot be added

It yields largest number of activities

∴ It is an optimal solution

(c) Earliest finish time :-

Start time	a[ ]	1	6	4	5	3	8	11	8	11	14
End time	b[ ]	2	7	7	8	9	10	12	12	13	15
Activity	Index	(1)	(2)	(7)	(8)	(3)	(4)	(5)	(6)	(9)	(10)

Selection of Activities = { 1, 2, 4, 5, 10 }

Activity 7 overlaps with 2 so cannot be added
Activity 8 overlaps with 2 so cannot be added
Activity 3 overlaps with 2 so cannot be added
Activity 6 overlaps with 4,5 so cannot be added
Activity 9 overlaps with 5 so cannot be added

It yields the largest number of activities

∴ It is an optimal Solution

Exercise 2: Solve and provide answer

☒ a:13 b:21 c:35 d:38 e:74 f:112 g:189

a) Fixed-length code of all characters:

a	b	c	d	e	f	g
000	001	010	011	100	101	110

b) Huffman code of all characters:

a	b	c	d	e	f	g
13	21	35	38	74	112	189

(Starting from the lowest value, add up the two lowest values together and keep going until only one remains. Also, don't forget to sort after adding so that all values appear in non-descending order.)

Let a = 0 and b = 1. Then a+b = 34.

ab	c	d	e	f	g
34	35	38	74	112	189

Let ab = 0 and c = 1. Then ab+c = 69.

d	abc	e	f	g
38	69	74	112	189

Let d = 0 and abc = 1. Then d+abc = 107.

e	dabc	f	g
74	107	112	189

Let e = 0 and dabc = 1. Then e+dabc = 181.

f	edabc	g
112	181	189

Let f = 0 and edabc = 1. Then f+edabc = 293.

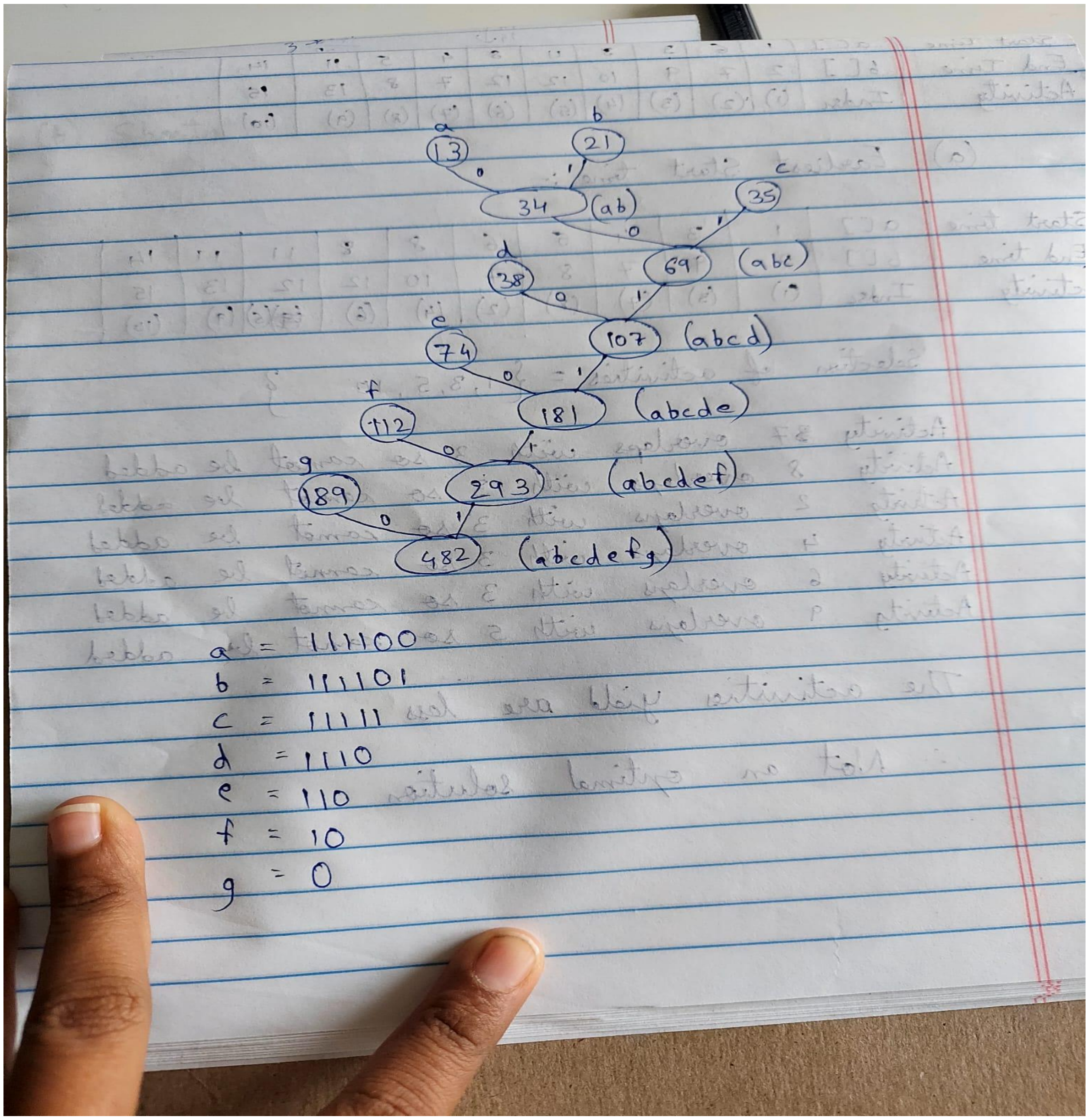
g	fedabc
189	293

Let g = 0 and fedabc = 1. Then g+fedabc = 482.

gfedabc
482

The resulting Huffman coding will be:

a	b	c	d	e	f	g
111100	111101	11111	1110	110	10	0



a = 111100
b = 111101
c = 11111
d = 1110
e = 110
f = 10
g = 0

Exercise 3: Solve and provide answer

```
Huffman(C)
Q.build(C)       //builds a min-priority queue on frequencies
for i ← 1 to n–1 do
    Allocate new node z
    x ← Q.extractMin() // Step 1 : O(1)
    y ← Q.extractMin() // step 1 : O(1)
    z.setLeft(x) //1
    z.setRight(y) //1
    z.setF(x.f() + y.f())
    Q.insert(z)
return Q.extractMin() // Return the root of the trie
```

Explanation: Symbol in the alphabets (frequency) are in the increasing order

Since a symbol in the alphabet are increasing order the smallest element can be found with O(1) for extractmin(). So, it just takes 1 step z.left for the first loop.

For the second extractmin(), smallest element can be found with O(1) for extractmin(). So, it just takes 1 step z.right to the second loop. The reason is that it is just making comparing with other minimum value.

If we insert element in to the front we are still going to make comparison between two lowest frequencies in the list. We can still make extraction, it will take 1 step for extractmin() for length=N.

Therefore, as an output, the algorithm can be O(n) with linear time complexity.