CSci 242: Algorithms and Data Structures **Spring, 2020**

Date: April 1st, 2020

**Due: by the end of day, April 10th (Fri.), 2020.**

**Home Assignment 7: Greedy Algorithm and Huffman Codes (105/100 points)**

**Q1 is void, so 25 points is given by default.**

**Q1. [VOID /25]** **Job Scheduling**

Each job takes one unit of time to finish and has a deadline and a profit. If the job starts before or at its deadline, the profit is obtained. The goal is to schedule the jobs so as to maximize the total profit. Not all jobs have to be scheduled. We need not consider any schedule that has one that doesn’t schedule the job at all.

A sequence is called a feasible sequence if all the jobs in the sequence start by their deadline. A set of jobs is called a feasible set if there exists at least one feasible sequence for the jobs in the set.

So, our goal is to find a ***feasible sequence of jobs with the maximum total profit.*** We call such a sequence an ***optimal sequence*** and the set of jobs in the sequence an ***optimal set of jobs***.

There are seven jobs with their deadline and profit:

|  |  |  |
| --- | --- | --- |
| Job | Deadline | Profit |
| A | 3 | 25 |
| B | 1 | 20 |
| C | 1 | 30 |
| D | 3 | 15 |
| E | 1 | 35 |
| F | 2 | 10 |
| G | 3 | 40 |

1. [0/5] What is a rationale of a greedy choice? i.e. In which order do you choose a job?
   1. Selecting the local optimum
2. [1/10] Write an iterative greedy algorithm, **Schedule(Jobs, *n*),** where *n* is the number of jobs and Jobs is a table of *n* jobs with their name, deadline profit, that decides both the ***optimal sequence of jobs*** to maximize the total profit and the ***total profit.***

I = 1

Total = 0

Job = []

Schedule(Jobs, N)

Total = total + job profit

Return total, job

1. [0/10, optional] Write a recursive greedy algorithm for the same problem.

I = 1

Total = 0

Job = []

Schedule(Jobs, N)

While I <= N

Total = total + job profit

I = I + !

Return job, total

1. [0/10] (A) What is the optimal sequence of jobs in the above problem? and

(B) What is its maximum total profit?

**Q1B [0/20 (+10), optional] Implementation of Q1.**

Write the Python (or Java) program for Q1.(2) (and (3)) to find both the optimal sequence of jobs and the maximum total profit.

**Q2. [10/10] Prefix Code**

A code for a, b, c, d, e is given by ‘a=00’, ‘b=01’, ‘c=101’, ‘**d=*x*10**’, ‘e=*yz1*’, where *x, y, z* are either 0 or 1. Determine *x, y* and *z* so that the given code is a prefix code.

A < B < C < D < E

So C < D

101 < x10 …. this condition is only true when x = 1

D = 110

D < E

110 < yz1 … this is only true when y = 1, z = 1

E = 111

Thus x = 1 y = 1 z = 1

**Q3. [17/25] Fractional Knapsack Problem**

An edited book has 6 articles. The table shows the lengths of the articles and their importance,

where the scale of importance is 1(low) to 10(high). The book must be at most 150 pages long.

The problem is ***to edit the book*** so that **the overall importance is maximized.**

1. [20] Edit the book by *choosing articles* whose pages and importance are given in the table. i.e. Give
2. [7/8] the list of the chosen articles with their chosen number of pages in the chosen order
   1. 10 : C
   2. 20 : 5 # you mean F but wrote 5.
   3. 15 : A
   4. 40 : 6 # you mean B but wrote 6.
   5. 30 : D
   6. 35/60 : C # you mean E but wrote C.
3. [5/6] the importance for each chosen article and
   1. FIND IN TABLE # since E is taken with 35 pages out of 60 total importance of E would be 0.666 x 35 = 2.333.
4. [0/6] the total maximum importance of the edited book.
   1. 240

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Article** | **Importance of article** | **Pages** | **UNIT** | **Importance** |
| A | 3 | 15 | 3/15 = 0.2 | 3 |
| B | 6 | 40 | 4/60 = 0.067 | 6 |
| C | 4 | 10 | 4/10 = 0.4 | 1 |
| D | 3 | 30 | 3/30 = 0.1 | 5 |
| E | 4 | 60 | 4/60 = 0.15 | 4 |
| F | 5 | 20 | 5/20 = 0.25 | 2 |

#correct answer for this part is: 4/C + 5/F + 3/A + 6/B + 3/D + 4\*(35/60)/E = 23.333

1. [5/5] Consider a greedy rule for the above Fractional Knapsack Problem that selects the articles in non-decreasing order of pages. If the capacity of the knapsack is not exceeded, we take all of the pages. Otherwise, we take whatever portion of the article fills the knapsack and stop. Give an example to show that this greedy algorithm does not necessarily maximize the importance.

10, 4 /C + 15,3 /A + 20, 5 / F + 30,3/ D + 40, 6 / B + 35,4 \*35/60 / E = 150 pages and 23.33 importance

Supposed it is 100 pages and not 150

Then it would be :   
10,4 / C + 20,5 / F + 15,3 /A + 40,6 / B + 15, 3\*15/30 / D= 100 pages and 19.5 importance

When the greedy is still

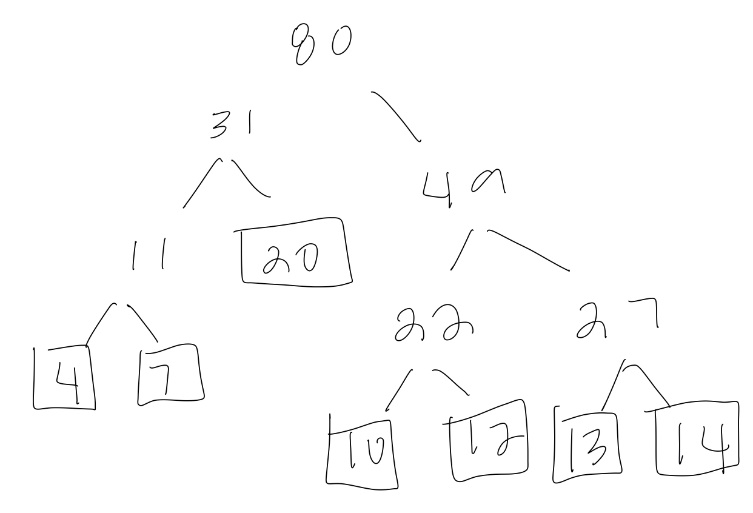
10,4 / C + 15,3 / A + 20,5 / F + 30,3 / D + 25, 6 \* 25/40 / B = 100 pages and 18.75 importance

**Q4. [35/40]** **Huffman Code**

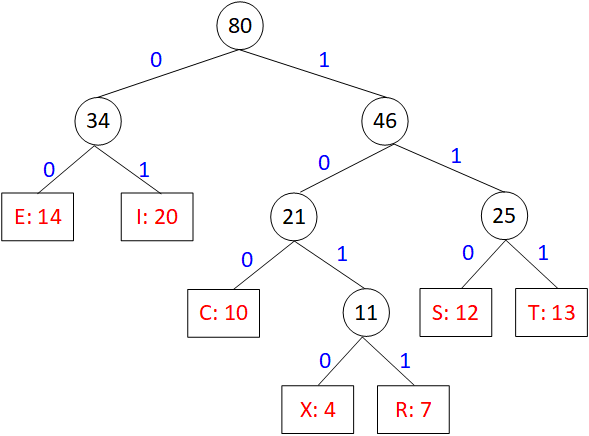
1. [20] Using Huffman Algorithm, construct an optimal binary prefix code for the letters in the given frequency table.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Letter | C | E | I | R | S | T | X |
| Frequency | 10 | 14 | 20 | 7 | 12 | 13 | 4 |
| (B) Code ? | 100 | 00 | 01 | 1011 | 110 | 111 | 1010 |

1. [3/8] Draw your Huffman Tree (i.e. Encoding Tree), and



# correct tree is as below:



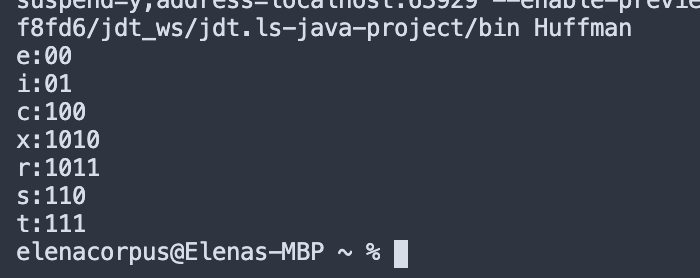
1. [7/7] Give an *optimal binary prefix code* for each letter.
   1. C : 100
   2. E : 00
   3. I : 01
   4. R : 1011
   5. S : 110
   6. T : 111
   7. X : 1010
2. [5/5] Give the *total number of bits* required to encode the string given in the frequency table, i.e. the total path weight
   1. 10 \* 3/ C + 14 \* 2 /E + 20 \* 2/ I + 7 \* 4 / R + 12 \* 3 / S + 13 \* 3 / T + 4 \* 4 / X
   2. = 30 + 28 + 40 + 28 + 36 + 39 + 16 = 217 BITS
3. [10/10] Decode each bit string using the Huffman code in 1.

Assumption: The Huffman tree is constructed with the left branches and the right branches which are labeled with 0 and 1, respectively.

1. 10010110011100
   1. CRETE
2. 1100000100110
   1. SEECS
3. 11100110111 001011
   1. TESTER
4. 101100110111
   1. REST
5. [10/10] Encode each word using the Huffman code in 1.
6. RISE
   1. 10110111000
7. EXIT
   1. 00101001111
8. TEXT
   1. 111001010111
9. EXERCISE
   1. 0010100010111000111000

**Q4B. [18/30, optional] Implementation of Q4.**

Write a Python/Java program of Huffman coding to

1. generate the optimal Huffman codes for the letters in Q4.(1) with their frequencies,
2. 
3. encode the words in Q4.(3) and
4. decode the bit strings in Q4.(2).