CSC 323 Algorithm Design and Analysis, Fall 2014
Instructor: Dr. Natarajan Meghanathan
Module 3 – Greedy Strategy, Question Bank

1) Show the binary representation of 173 using the greedy strategy discussed in the slides.

128	64	32	16	8	4	2	1
1	0	1	0	1	1	0	1

2) Determine the change that would incur the minimum number of coins for an amount of 47 cents. Use the standard coin denomination values used in the US.

25	10	5	1
1	2	0	2

47 cents = 1 quarter (25 cents) + 2 dimes (2*10 = 20 cents) + 2 pennies (2*1 cent = 2 cents)

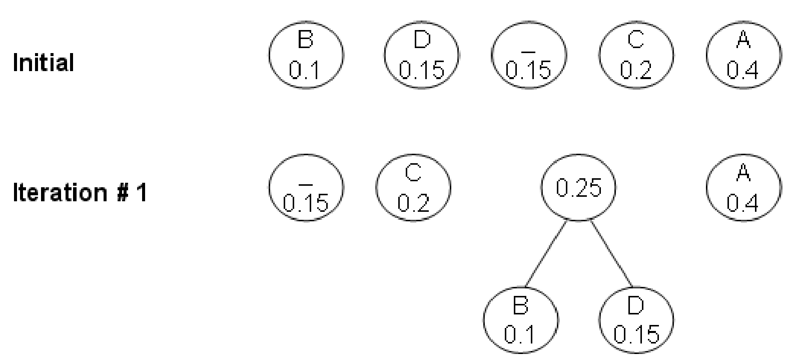
3) What is a prefix-free code? What is its advantage? Justify why Huffman's codes are prefix-free codes?

- In a prefix-free code, no codeword is a prefix of a code of another symbol. With a prefix-free code based encoding, we can simply scan a bit string until we get the first group of bits that is a codeword for some symbol, replace these bits by this symbol, and repeat this operation until the bit string's end is reached.
- With the Huffman algorithm-based encoding, the binary codes are assigned based on a simple path traversed from the root to a leaf node representing the symbol. Since there cannot be a simple path from the root to a leaf node that leads to another leaf node (then we have to trace back some intermediate node – meaning a cycle). Hence, Huffman codes are prefix-free codes.

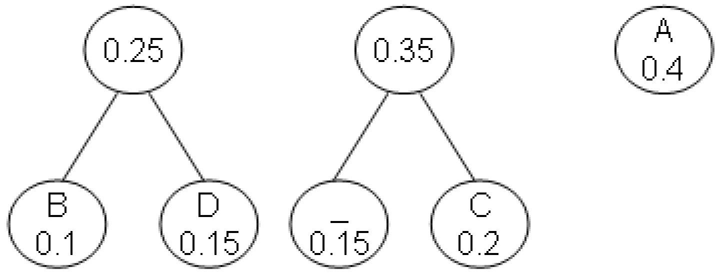
4) Construct a Huffman code for the following data (show all the steps):

symbol	A	B	C	D	_
frequency	0.4	0.1	0.2	0.15	0.15

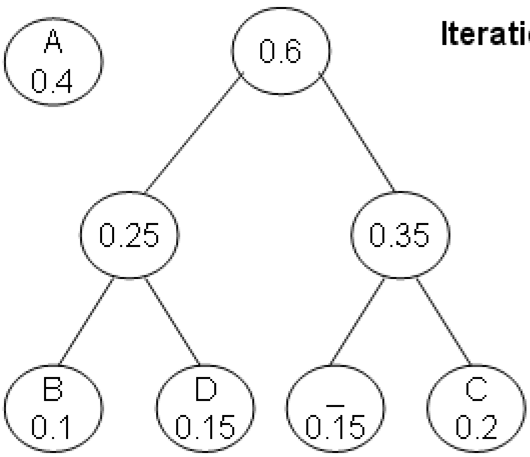
(a) Determine the average number of bits per symbol.
(b) Determine the compression ratio (generic) compared to fixed-length encoding.
(c) Encode ABACABAD using the Huffman code that you determined. Determine the compression ratio achieved for this text compared to fixed-length encoding.
(d) Decode 100010111001010 using the Huffman code that you determined.



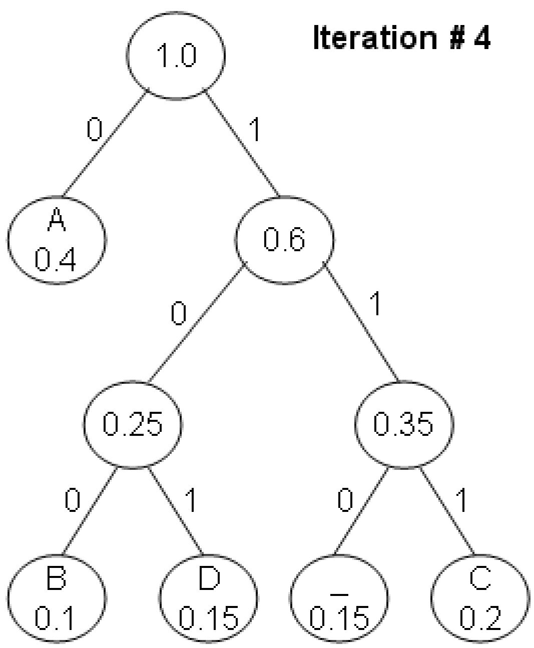
Iteration # 2



Iteration # 3



Iteration # 4



A	0.4	0
B	0.1	100
C	0.2	111
D	0.15	101
_	0.15	110

Avg. # bits per symbol
= 0.4*1 + 0.1*3 + 0.2*3 + 0.15*3 + 0.15*3
= 2.2 bits

Fixed length encoding would require $\lceil \log_2 5 \rceil$= 3 bits

Hence, the generic compression ratio = 1 – (2.2/3)
= 26.7%

(c)
ABACABAD

0 100 0 111 0 100 0 101

The total number of bits in the binary code for the given 8-symbol text is 16 bits. A fixed-length encoding would have result in 8*3 = 24 bits. Hence, the compression ratio that is specific for this text is: 1 – (16/24) = 33.3%.

(d)

100010111001010

100010111001010	100	B
100010111001010	0	A
100010111001010	101	D
100010111001010	110	_
100010111001010	0	A
100010111001010	101	D
100010111001010	0	_

The decoded text is BAD_AD_

5) Solve the following instances of the Knapsack problem as a <u>Fractional Knapsack problem</u>

(a) Knapsack weight = 5 lb.

Item	1	2	3	4
Value, $	12	10	20	15
Weight, lb	2	1	3	2

Solution: Compute the Value/Weight for each item

Item	1	2	3	4
Value/Weight	6	10	6.67	7.5

Re-ordering the items according to the decreasing order of Value/Weight (break the tie by picking the item with the lowest Index)

Item	2	4	3	1
Value/Weight	10	7.5	6.67	6
Value, $	10	15	20	12
Weight, lb	1	2	3	2
Weight collected	1	2	2	

Items collected: Item 2 (1 lb, $10); Item 4 (2 lb, $15); Item 3 (2 lb, (2/3)*20 = $13.3);

Total Value = $38.3

(b) Knapsack weight = 6 lb.

Item	1	2	3	4	5
Value, $	5	8	10	6	4
Weight, lb	2	3	4	1	2

Solution: Compute the Value/Weight for each item

Item	1	2	3	4	5

Value/Weight	2.5	2.7	2.5	6	2

Re-ordering the items according to the decreasing order of Value/Weight (break the tie by picking the item with the lowest Index)

Item	4	2	1	3	5
Value/Weight	6	2.7	2.5	2.5	2
Value, $	6	8	5	10	4
Weight, lb	1	3	2	4	2
Weight collected	1	3	2		

Items collected: Item 4 (1 lb, \$6); Item 2 (3 lb, \$8); Item 1 (2 lb, \$5)
Total Value = \$19

6) Given the following list of activities, find the list of activities for maximal conflict-free scheduling.

Activity	1	2	3	4	5	6	7	8	9	10
Start	1	1	2	4	5	8	9	11	12	13
Finish	3	8	5	7	9	10	11	14	17	16

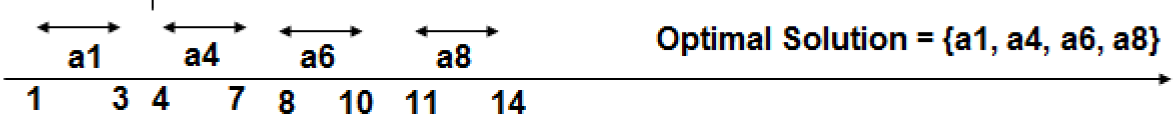
Solution:

Sorted List (increasing order of finish time)

Activity	1	3	4	2	5	6	7	8	10	9
Start	1	2	4	1	5	8	9	11	13	12
Finish	3	5	7	8	9	10	11	14	16	17

Sorted List (Selected/ Discarded Activities)

Activity	1	~~3~~	4	~~2~~	~~5~~	6	~~7~~	8	~~10~~	~~9~~
Start	1	~~2~~	4	~~1~~	~~5~~	8	~~9~~	11	~~13~~	~~12~~
Finish	3	~~5~~	7	~~8~~	~~9~~	10	~~11~~	14	~~16~~	~~17~~



Optimal Solution = {a1, a4, a6, a8}

7) Prove the following theorems with respect to the Minimal Finish Time strategy for Activities Selection problem.

Theorem 1: At least one maximal conflict-free schedule includes the activity that finishes first.
Proof (by contradiction): There may be several maximal conflict-free schedules.

- But, assume the activity finishing first (say *u*) is in none of them.
- Let X be one such maximal conflict-free schedule that does not include *u*. Let *v* be the activity finishing first in X.

- Since u finishes before v, u should not conflict with activities X – {v}.
- Hence, v could be removed from X and u could be inserted to X, leading to X' = X U {u} – {v}.
- The set X' featuring u would also be a maximal conflict-free schedule.

Theorem 2: The greedy schedule formed based on the earliest finishing activities is optimal.
Proof:

- Let u be the earliest finishing activity. According to Theorem 1, u will be part of some maximal conflict-free schedule X.
- Since u is the earliest finishing activity, it should be the first activity in X.
- Among all the activities that overlap with u in X, only one of them could be selected for X (in this case, u is indeed selected for X).
- Let Y = X – {u} – {set of all activities overlapping with u}. The optimality of the conflict-free schedule for Y will hold true due to induction.

8) Based on each of the following criteria, determine the order in which the following files should be organized in a tape to minimize the average access time. Determine the average access time in each case.

File Index	1	2	3	4	5	6	7	8
File Size	10	15	5	20	45	12	25	18
Acc. Frequency	5	10	8	7	9	6	12	13

(i) Increasing order of file index
(ii) Increasing order of file size
(iii) Increasing order of file size / access frequency

Solutions:

(i) Increasing order of file index

Sorting based on the increasing order of File Index only

File Index	1	2	3	4	5	6	7	8
File Size	10	15	5	20	45	12	25	18
Acc. Frequency	5	10	8	7	9	6	12	13
Cost to Access	10	25	30	50	95	107	132	150
Cost*Freq	50	250	240	350	855	642	1584	1950

$$\text{Average cost to access any file} = \frac{(50 + 250 + 240 + 350 + 855 + 642 + 1584 + 1950)}{(5 + 10 + 8 + 7 + 9 + 6 + 12 + 13)}$$

$$= 84.58$$

(ii) Increasing order of file size

Sorting based on the increasing order of File Size only

File Index	3	1	6	2	8	4	7	5
File Size	5	10	12	15	18	20	25	45
Acc. Freq.	8	5	6	10	13	7	12	9
Cost to Access	5	15	27	42	60	80	105	150
Cost*Freq	40	75	162	420	780	560	1260	1350

$$\text{Average cost to access any file} = \frac{(40 + 75 + 162 + 420 + 780 + 560 + 1260 + 1350)}{(8 + 5 + 6 + 10 + 13 + 7 + 12 + 9)}$$

$= 66.38$

(iii) Increasing order of file size / access frequency

Sorting based on the increasing order of File Size / Access Frequency

File Index	3	8	2	1	6	7	4	5
File Size	5	18	15	10	12	25	20	45
Acc. Freq.	8	13	10	5	6	12	7	9
Size/Frequency	0.625	1.385	1.5	2	2	2.083	2.857	5
Cost to Access	5	23	38	48	60	85	105	150
Cost*Freq	40	299	380	240	360	1020	735	1350

$$\text{Average cost to access any file} = \frac{(40 + 299 + 380 + 240 + 360 + 1020 + 735 + 1350)}{(8 + 13 + 10 + 5 + 6 + 12 + 7 + 9)}$$

$= 63.2$

9) Prove that the strategy of ordering the files in the increasing order of File Size / Access Frequency gives the optimal solution for the Tape Read Scheduling problem.

- We want to prove that we get an optimal solution, when:

$$\boxed{\frac{L[\pi(i)]}{F[\pi(i)]} \leq \frac{L[\pi(i+1)]}{F[\pi(i+1)]} \text{ for all } i}$$

- Where, π(i) is the position of File i in the sorted order of Size/Frequency; L[π(i)] is the length of File i.
- Suppose L[π(i)] / F[π(i)] > L[π(i+1)] / F[π(i+1)] for some i.
- To simplify notation, let a = π(i) and b = π(i+1). L[a]/F[a] > L[b]/F[b]
- If we swap files a and b, then the cost of accessing a increases by L[b] and the cost of accessing b decreases by L[a]. Overall, the swap changes the total cost by L[b]F[a] – L[a]F[b] < 0. This is an improvement! We do this for all consecutive pairs a and b.