__/__/____

ADA

Greedy Algorithm

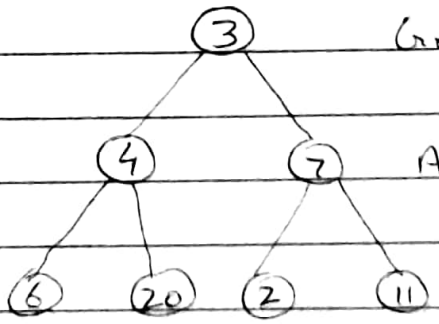
A greedy algo is a Strategy that makes the optimal choice at each stage with hope of finding a global optimal.

Pros: Simple, easy to implement, run fast.
Cons: do not always yield optimal Solution.

Characteristics of greedy algorithm

1) used to solve optimization problem.
2) Most general, straight forward method to solve a problem.
3) Always makes the choice that looks best at the moment i.e it makes a "locally optimal choice in the hope that this choice will lead to a overall globally optimal Solution".
4) It does not always yields an optimal Solution but for many problem they do.
5) Once any choice of input from C is rejected then it never considered again.

Eg:



Minimization

Greedy:
3 + 4 + 6 = 13

Actual:
3 + 7 + 2 = 12

Maximization:

Greedy: 3 + 7 + 11 = 21
Actual: 3 + 4 + 20 = 27

Time complexity = $n \log n + n = (n \log n)$

__/__/____

→ Activity Selection Problem
→ Huffman Coding
→ Knapsack Problem
→ finding Minimum Spanning tree.
  ↳ Krushkal's algo
  ↳ Prim's algo
→ Single Source Shortest path
  ↳ Dijkstra's algo
  ↳ Bellmenford algo

⇒ Activity Selection Problem

There are n different activity are given (with their Starting and ending time. Select maximum number of activity to solve by a single person.

1) Sort the activity with their ending time.
2) Find compatible activity and add to list.

Example:

A_i	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9
S_i	1	2	4	1	5	8	9	11	13
T_i	3	5	7	8	9	10	11	14	16

$n = 9$

Greedy approach : $A = \{A_1, A_3, A_6, A_8\}$

General approach : $A = \{A_2, A_5, A_7, A_8\}$

Total 4 activity.

__/__/__

Algorithm:

Greedy-activity (S, f)

1) $n \leftarrow$ Length (S)
2) $A \leftarrow \{1\}$
3) $J \leftarrow 1$
4) for $i \leftarrow 2$ to n
5) do if $S_i \geq f_J$
6) then $A \leftarrow A \cup \{i\}$
7) $J \leftarrow i$
8) Return A

Q Given 10 activity along with their start & finish time.

$S = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10}\}$

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
S_i =	1	2	3	4	7	8	9	~~10~~ 9	11	12
f_i =	3	5	4	7	10	9	11	13	12	14

loc:	1	2	3	4	5	6	7	8	9	10
S_i	1	3	2	4	8	7	9	11	9	12
f_i	3	4	5	7	9	10	11	12	13	14
	A_1	A_3	A_2	A_4	A_6	A_5	A_7	A_9	A_8	A_{10}

$n = 10$

__/__/____

Huffman Coding

Huffman Coding used to compress the data upto 90%. Suppose we've 100 character data having six different character.

	A	B	C	D	E	F	
Freq	45	13	12	16	9	5	=100

ASCII 000100.....001001010..... = 100 x 8 bit = 800 bit

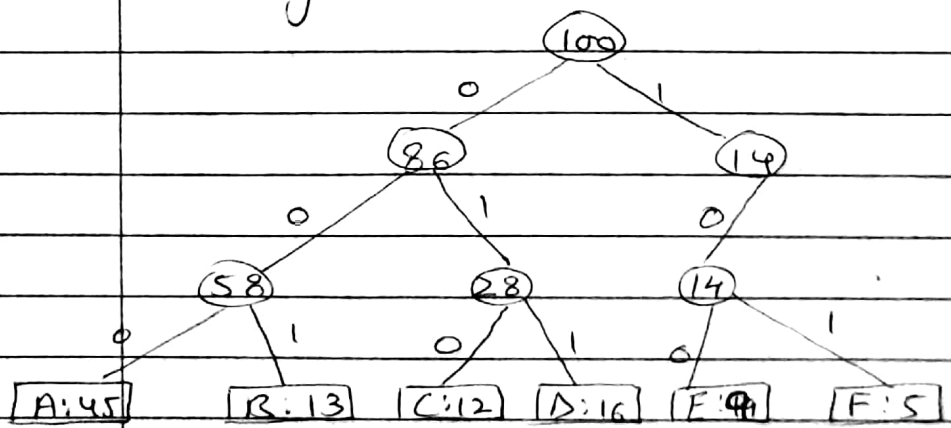
fix length = 000 001 010 011 100 101 = 100 x 3 = 300 bit

van length = 0 101 100 111 1101 1100

= 45x1 + 13x3 + 12x3 + 16x3 + 9x4 + 5x4

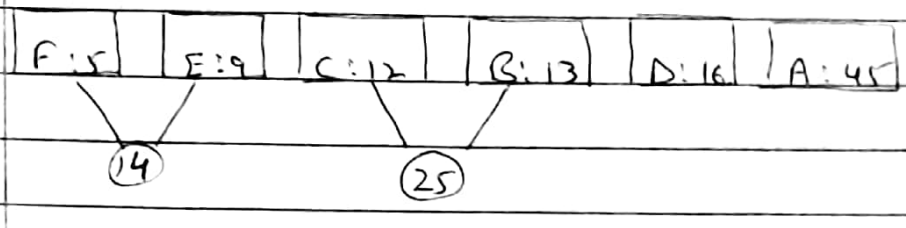
= 224 bit

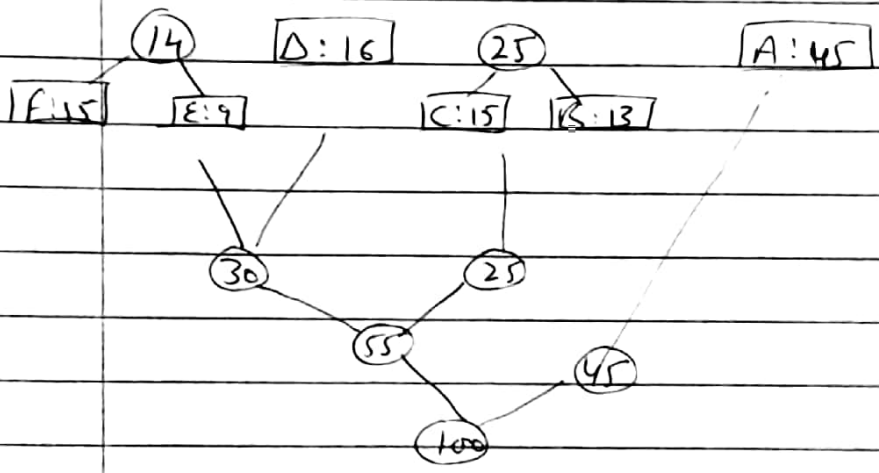
Fixed length

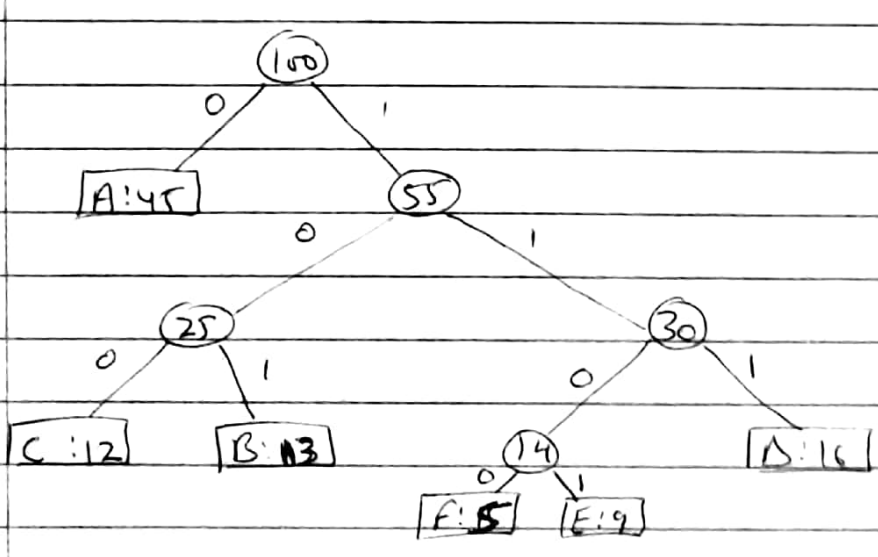


= 300 + 6x8 + 18

= 300 + 48 + 18 = 366 bit







VL

= 224 bit

= 224 + 48 + 18 = 290 bit

Note: Huffman coding use variable length.

__/__/____

Knapsack Problem (Fraction)

We have n object and a knapsack (bag). Each object have some profit and weight. The capacity of knapsack (bag) is m.

Problem: Fill the bag with object that maximize the profit.

Example: $n = 3 \qquad m = 20$

$(P_1, P_2, P_3) = (25, 24, 15)$

$(w_1, w_2, w_3) = (18, 15, 10)$

Solution:

	x_1	x_2	x_3
Object	1	2	3
Profit	25	24	15
Weight	18	15	10
Profit/weight	1.38	1.6	1.5

	x_1	x_2	x_3	$\sum w_i x_i$	$\sum p_i x_i$
Max Profit:	1	$\frac{2}{15}$	0	$18 + \frac{2}{15} \times 15 = 20$	$25 \times 1 + 24 \times \frac{2}{15} = 28.2$
Min weight	0	$\frac{10}{15}$	1	$10 \times 1 + \frac{10}{15} \times 15 = 10 + 10 = 20$	$15 \times 1 + 24 \times \frac{10}{15} = 31$
Profit/weight (Max)	0	1	$\frac{5}{10}$	$15 \times 1 + \frac{5}{10} \times 10 = 20$	$24 \times 1 + \frac{5}{10} \times 15 = 31.5$

$\Rightarrow$ Profit/weight gives optimal solution.