gradeup

Design Techniques Part-3

Design Technique Part-3

Content:

1. **Huffman Codes**
2. **Travelling Salesman Problem**

HUFFMAN CODES

Data can be encoded efficiently using Huffman codes. It is used and effective technique for compressing data; savings of 20% to 90% are typical, depending on the characteristics of the file being compressed. It uses a table of the frequencies of occurrence of each character to build up an optimal way of representing each character as a binary string.

Suppose we have 10^5 characters in a data file. Normal storage: 8 bits per character (ASCII)- 8×10^5 bits in file. But, we want to compress the file and store it compactly. Suppose only 6 characters appear in the file:

	a	*b*	*c*	*d*	*e*	*f*	Total
Frequency	45	13	12	16	9	5	100

How can we represent the data in a compact way?

i. **Fixed Length code.** Each latter represented by an equal number of bits. With a fixed length code, at least 3 bits per character:
For example:

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

for a file with 10^5 characters, we need 3×10^5 bits.

ii. **A variable-length code** can do considerably better than a fixed-length code, by giving frequent characters short code words and infrequent characters long code words.
For example:

a	0
b	101
c	100
d	111
e	1101
f	1100

number of bits = (45*1 + 13*3 + 12*3 + 16*3 + 9*4 + 5*4)*1000

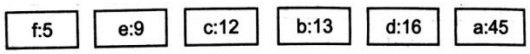

Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

$= 2.24 \times 10^5$ bits.

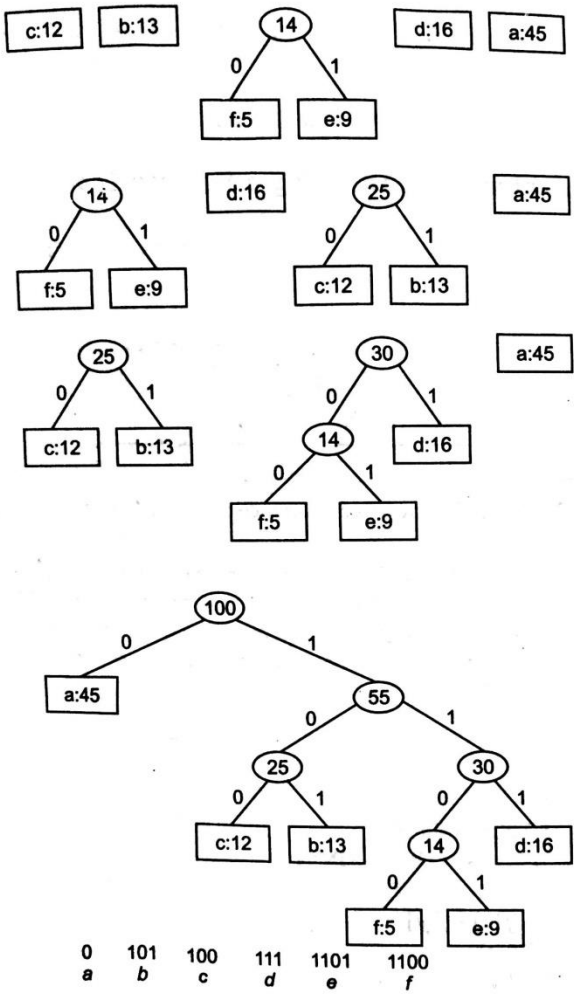
Thus, 224000 bits to represent the file, a saving of approximately 25%. This is an optimal character code for this file.

The analysis of the running time of Huffman's algorithm assumes that Q is implemented as a binary heap. For a set C of n characters, the initialization of Q in line 2 can be performed in O(n) time using the BUILD-HEAP procedure. The for loop in lines 3-8 is executed exactly $|n|-1$ times, and since each heap operation requires time O(n lg n), the loop contributes O(n lg n) to the running time. Thus, the total running time of HUFFMAN on a set of n character is O(n lg n).

Example:



The algorithm is based on a depletion of a problem with n characters to a problem with n – 1 characters. A new character replaces two existing ones.



Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

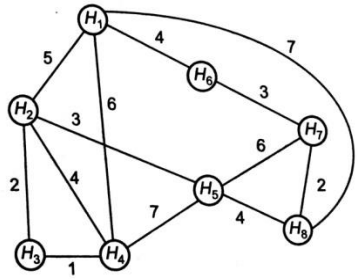
TRAVELLING SALES PERSON PROBLEM

In this a salesman needs to visit 'n' cities in such a manner that all cities must be visited at once and in the end he returns to the city from where he started with minimum cost.

Suppose the cities are x_1, x_2 …. x_n where c_{ij} denotes the cost of travelling from city x_1. The method is represented until all the cities are visited and at every step we have to select the city with the minimum weight.

Example: A newspaper agent daily drops the newspaper to the area assigned in such a manner that he has to cover all the houses in the respective area with minimum travel cost. Compute the minimum travel cost.

The area assigned to the agent where he has to drop the newspaper is shown in below figure.



Solution: The cost-adjacency matrix of graph G is as follows:

$Cost_{ij}$=

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	4	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	3	0
H_7	0	0	0	0	6	3	0	2
H_8	7	0	0	0	4	0	2	0

The tour starts from area H_1 and then selected the minimum cost area reachable from H_1.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	4	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	3	0
H_7	0	0	0	0	6	3	0	2
H_8	7	0	0	0	4	0	2	0


Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

Mark area H_6 because it is the minimum cost area reachable from H_1 then select minimum cost area reachable from H_6.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	(4)	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	(3)	0
H_7	0	0	0	0	6	3	0	2
H_8	7	0	0	0	4	0	2	0

Mark area H_7 because it is the minimum cost area reachable from area H_6 and then select minimum cost area reachable from H_7.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	(4)	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	(3)	0
H_7	0	0	0	0	6	3	0	(2)
H_8	7	0	0	0	4	0	2	0

Mark area H_8 and select minimum cost area reachable from H_8.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	(4)	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	(3)	0
H_7	0	0	0	0	6	3	0	(2)
H_8	7	0	0	0	(4)	0	2	0

Mark area H_5 and select minimum cost area reachable from H_5.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	(4)	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	(3)	0	7	0	0	6	4
H_6	4	0	0	0	0	0	(3)	0
H_7	0	0	0	0	6	3	0	(2)
H_8	7	0	0	0	(4)	0	2	0


Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

Mark area H_2 and select minimum cost area reachable from H_2.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	(4)	0	7
H_2	5	0	(2)	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	(3)	0	7	0	0	6	4
H_6	4	0	0	0	0	0	(3)	0
H_7	0	0	0	0	6	3	0	(2)
H_8	7	0	0	0	(4)	0	2	0

Mark area H_3 and select minimum cost area reachable from H_3.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	(4)	0	7
H_2	5	0	(2)	4	3	0	0	0
H_3	0	2	0	(1)	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	(3)	0	7	0	0	6	4
H_6	4	0	0	0	0	0	(3)	0
H_7	0	0	0	0	6	3	0	(2)
H_8	7	0	0	0	(4)	0	2	0

Mark area H_4 and select minimum cost area reachable from H_4 it is H_1.

So, using the greedy strategy we get the following.

$$H_1 \overset{4}{\rightarrow} H_6 \overset{3}{\rightarrow} H_7 \overset{2}{\rightarrow} H_8 \overset{4}{\rightarrow} H_5 \overset{3}{\rightarrow} H_2 \overset{2}{\rightarrow} H_3 \overset{1}{\rightarrow} H_4 \overset{6}{\rightarrow} H_1$$

Thus, the minimum travel cost

= 4+3+2+4+3+2+1+6 = 25

BRANCH AND BOUND ALGORITHMS

They are generally used for optimization problem. As the algorithms progresses, a tree of sub-problems is formed. The original problem is considered the “root problems”.

A process is used to construct an upper and lower bound for a given problem.

- At each node, apply the bounding methods.
- If the bound match, it is deemed a feasible solution to that particular sub problem.
- If bounds do not match, partition the problem represented by that node, and make the two sub problems into children nodes.


Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

- Continue, using the best known feasible solution to trim sections of the tree, until all nodes have been solved or trimmed.

BRUTE FORCEF ALGORITHMS

A brute force algorithm simply tries all possibilities until a satisfactory solution is found. Such an algorithm can be:

- **Optimizing.** Find the best solution. This may require finding all solution, or if value for the best solution is known, it may stop when any best solution is found. Example: Finding the best path for a travelling salesman.
- **Satisfying.** Stop as soon as a solution is found that is good enough. Example, finding a travelling salesman path that is within 10% of optimal.

RANDOMIZED ALGORITHMS

It uses a random number at least once during the computation to make a decision.

- Example, In Quick sort, using a random number of choose a pivot.
- Example, Trying to factor a large number by choosing random numbers as possible divisors.


Gradeup UGC NET Super Subscription
Access to all Structured Courses & Test Series

gradeup SUPER

Gradeup UGC NET
Super Superscription

Features:

1. 7+ Structured Courses for UGC NET Exam
2. 200+ Mock Tests for UGC NET & MHSET Exams
3. Separate Batches in Hindi & English
4. Mock Tests are available in Hindi & English
5. Available on Mobile & Desktop

Gradeup Super Subscription, Enroll Now

gradeup.co