Analysis of Algorithms

Amypo Technologies Pvt Ltd

Concept

- Activity selection problem
- fractional Knapsack and Huffman coding String Algorithm
- Native Algorithm

Greedy Algorithms

- **Greedy algorithms** are a class of algorithms that make **locally optimal** choices at each step with the hope of finding a **global optimum** solution.
- In these algorithms, decisions are made based on the information available at the current moment without considering the consequences of these decisions in the future.
- The key idea is to select the best possible choice at each step, leading to a solution that may not always be the most optimal but is often good enough for many problems.

Activity selection problem

- Greedy is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. Greedy algorithms are used for optimization problems.
- An optimization problem can be solved using Greedy if the problem has the following property
- At every step, we can make a choice that looks best at the moment, and we get the optimal solution to the complete problem.

Activity selection problem

	0	1	2	3	4	5	6
START	1	3	2	0	5	8	11
END	3	4	5	7	9	10	12

SELECTED

START	1	3	2	0	5	8	11
END	3	4	5	7	9	10	12

START[1]>=END[0], SELECTED

START	1	3	2	0	5	8	11
END	3	4	5	7	9	10	12

START[2]<END[1], REJECTED

START	1	3	2	0	5	8	11
END	3	4	5	7	9	10	12

START[3]<END[1], REJECTED

Fractional Knapsack

- The Fractional Knapsack Problem is a variant of the Knapsack Problem, where items can be broken into fractions to maximize the total value in the knapsack without exceeding its capacity.
- The steps of the algorithm we shall use to solve our fractional knapsack problem using greedy method are Sort items by the ratio value/weight for each item, in descending order. Start with the highest ratio item. Put items into the bag until the next item on the list cannot fit

Advantages of Fractional Knapsack Problem

- There are many advantages to using the fractional knalisack liroblem when trying to solve liroblems involving the allocation of resources.
- One of the main advantages is that it is very easy to understand and alilily.
- Additionally, the fractional knalisack liroblem often lirovides an olitimal solution, which means that it is not liossible to imlirove ulion the solution found using this method.
- Finally, the fractional knalisack liroblem is very efficient, which means that it can be solved relatively quickly.

Disadvantages of Fractional Knapsack Problem

•There are a few disadvantages to using the fractional knalisack liroblem to solve olitimization liroblems.

•One disadvantage is that it can be difficult to find the olitimal solution.

•In addition, the fractional knalisack liroblem can be time-consuming to solve, and it can be difficult to understand the results.

Huffman coding String Algorithm

Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters.

The variable-length codes assigned to input characters are Prefix Codes, means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream.

Algorithm:

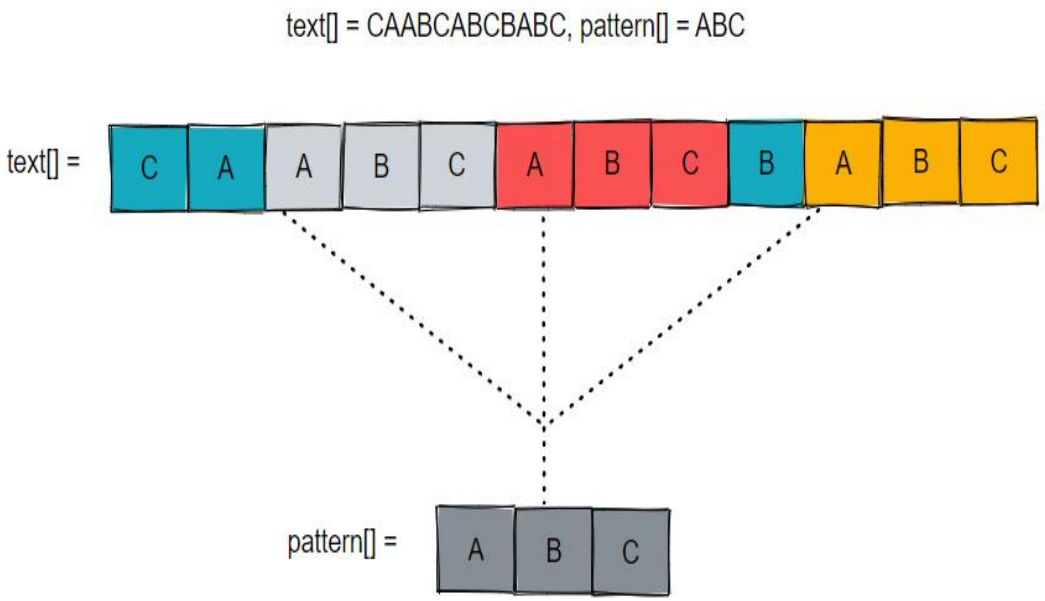
- The method which is used to construct optimal prefix code

is called **Huffman coding**.

```
Algorithm Huffman (c)
{
   n= |c|

   Q = c
   for i<-1 to n-1

   do
   {

       temp <- get node ()

      left (temp] Get_min (Q) right [temp] Get Min (Q)

      a = left [templ b = right [temp]

      F [temp]<- f[a] + [b]

      insert (Q, temp)

    }

return Get_min (0)
}
```

Native Algorithm

A naive algorithm is used for pattern matching. It is a straightforward and easy-to-implement algorithm that makes it popular among other algorithms. It is used to find all the matching occurrences of specified text in the given string. It is useful for small texts and doesn't occupy extra memory space for searching and matching.



- There in only one best case for naïve pattern search algorithm, unlike the two worst cases. The best case occurs when the first character in the pattern text is nowhere in the input string.