Gunjan
Date
Page

DAA Mid term II

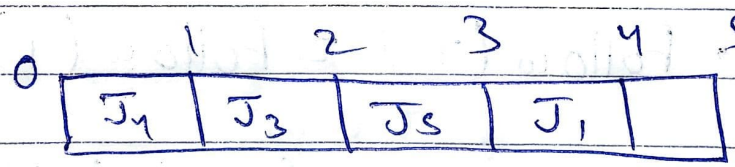
SET-A

Ans 1. Greedy method :-

Greedy Algorithms makes good local choices in the hope that they result in an optional solution:-

- They results in feasible solution.
- Not necessarily an optimal solution.
- A proof is needed to show that the algorithm finds an optimal solutions.

Jobs	J1	J2	J3	J4	J5
Profits	30	12	5	20	15
Deadlins	4	3	2	1	3



Max profit = 30 + 20 + 15 + 5 = (J4 + J3 + J5 + J

= 70.

Ans 2.

In dynamic Programming approach, we always break down the main problem into several smaller subproblems.

Later the solutions to those subproblems are combined to give the solution to a main problem.

Steps to develop a dynamic programming algorithm :-

1. Characterize the structure of an optimal sol^n.
2. Recursively define the value of an optimal sol^n.
3. Compute the value of an optimal sol^n. typically in a bottom-up manner.
4. Construct an optimal sol^n from computed information.

* Matrix Chain Multiplication :- We have sequence (chain) $\langle A_1, A_2, \ldots A_n \rangle$ of n matrices to be multiplied & we wish to compute the product. is

A product of matrix is fully parenthesized if it is either a single matrix of the product of two fully parenthesized matrix surrounded by parenthesis, matrix multiplication

is associative & so all parenthesis yield all the same product

e.g $1 \le k < n-1$, then

$$A_{1..n} + A_{1..k} \times A_{k+1..n}$$

$$= \begin{cases} 0 & \text{if } i = j \\ \min(m\,mul[i,k] + m\,mul[k+1,j] + p_{k-1} p_k p_j & \text{if } 1 \le j, i \le k < j \end{cases}$$

Ans 3. Fractional knapsack - In this, the list of items are divisible ; that means we can take any fraction of an item. We describe the problem using Greedy approach.

0/1 Knapsack Problem - In this problem the list of item are indivisible ; that means either we take the item or discard it. We will disscuss this problem using DP approach.

Also we can say that → In "Fractional knapsack" we can take objects in fraction.

And in "0/1 knapsack" we can take the objects in whole number.

Gunjan
Date
Page

Ans 4. In Brute-Force-Algorithm, we have to backup the text pointer for every mismatch. Some character are examined twice or thrice depending upon the shifts for the given pattern.

The finite automata is used to eliminate the needless shifts, but the complicated preprocessing for ~~every mismatch. Some characters~~ the computation of transistion fun' pays off.

The KMP algorithms has a linear running time of $O(n+m)$, which is achieved by using the auxillary fun' Pf (prefix fun') ,pre-processed from pattern P in time $O(m)$.

```
KMP Matcher (T, P)
1.  n ← length [T]
2.  m ← length [P]
3.  π ← COMPUTE-PREFIX-FUNCTION (P)
4.  q ← 0                    ▷ No. of characters matched.
5.  for i ← 1 to n           ▷ Scan the text from left to
6.  do-while q > 0 & P[q+1] ≠ T[i]
7.  do q ← π [q]
8.  if P[q+1] = T[i]
9.      then q ← q+1.
10.     if q = m
11.     then print "Pattern occurs with shift" i-m
12.     q ← π [q].
```

Ans 5.

Longest Common Subsequence

↳ The dynamic Programming (DP) technique suggests a breakup of the main problem into several smaller subproblems.

A prefix of a sequence indicates an initial string of values, $P_i = \langle P_1, P_2, P_3 \ldots P_i \rangle$. P_0 indicates the empty sequence. The method is related to compute the LCS for all possible pair of prefixes. Consider LCS [i, j] to be the length of the longest common subsequence of P_i & q_j for example.

$P_3 = \langle KLO \rangle$

$q_4 = \langle XKLL \rangle$

Thus, their longest common subsequence LCS is $\langle KL \rangle$.

Therefore LCS [3, 4] = 2.

The DP method is to compute LCS [i, j] in a manner by assuming that LCS [i, j] already computed where $i' < i$ & $j' \leq j$ but not equal.

(1) Basis, Either of the sequence is empty
(2) When last character of both the sequence
(3) When Last character of either sequence do not match: