Important Que (0/1 Knapsack, TSP, String Editing Problem)

Chapter 4 (2Q)
Dynamic Programming

1 1) Briefly explain the dynamic programming method for problem solving. What is the basic difference between Dynamic programming and Greedy method

Ans. Dynamic programming is an ~~optimal optimar~~ optimization method which is used to find the optimal solution of a problem.

→ It is a technique for solving a complex problem by first breaking into a collection of simplex subproblems, solving each subproblem just once, and then storing their solutions for future use to avoid repetitive computations.

→ (Description in detail:-

Dynamic programming is a technique that breaks the problems into-subproblems and saves the result for future purposes so that we do not need to compute ~~the~~ the result again. The main use of dynamic ~~problem~~ programming is to solve optimization problems. Here optimization ~~recent~~ problems mean that we are trying to find out the minimum or the maximum solution of a problem. The dynamic programming guarantees to find the optimal solution of a problem if the solution exists.

→ In the case of dynamic programming, the space complexity would increase as we will store the intermediate results,

Scanned with CamScanner

but the time complexity will decrease.

Steps in dynamic programming:

1) It breaks down the complex problem into simpler subproblems.
2) It finds the optimal solution of these sub-problems.
3) It stores the results of subproblems (memorization).
4) It ~~reuses~~ reuses the results if the subproblem should be calculated more than once.
5) Finally, calculate the result of the complex problem.

Greedy Algorithm vs Dynamic Programming

-> Greedy Algorithms are similar to dynamic programming in the sense they are both tools for optimization.

-> However, greedy algorithms look for locally optimal solutions i.e greedy choice in the hopes of finding a global optimal solution. Hence, greedy algorithm can make a guess that looks optimal at the time but becomes costly down the line and do not guarantee a global optimal solution.

-> On the other hand, it is guaranteed that dynamic programming will generate an optimal solution as it generally considers all possible cases and then choose the best.

Scanned with CamScanner

classmate
Date
Page

Different defination of Dynamic programming:

(In dynamic programming a problem is given. Then we perform brute force approach to find all the possible solutions. Then the solution which is optimal is choosen.)

Greedy method	Dynamic Programming
1) In a greedy algorithm, we make whatever choice seems best at the moment in the hope that it will lead to global optimal solution.	1) In dynamic programming, we make decision at each step considering current problem and solution to previously solved sub problem to calculate optimal solution.
2) There is no guarantee of getting optimal solution.	2) It is guaranteed to generate an optimal solution using Principle of optimality.
3) It is more efficient interms of memory because it never looks back or revise previous choices.	3) It requires memorization table which increases it's memory complexity.
4) Eg: fractional knapsack	4) 0/1 knapsack problem

Scanned with CamScanner

2) Compare Divide and Conquer and dynamic programming approaches of algorithm design with examples.

Ans: In Divide and Conquer method, the problem is divided into small subproblems. These subproblems are solved independently. Finally, all the solutions of subproblems are collected together to get the solution to the given problem.

→ It deals three steps at each level of recursion

i) Divide: the problem into a number of subproblems.

ii) Conquer the subproblems by solving them recursively.

~~iii) Combine the solution of subproblems into the solution for original problem.~~

iii) Combines the solution of subproblems to find the solution to the original problem.

→ ~~However~~, In dynamic programming it breaks the problem into sub-problems like in divide and conquer method. However, it doesnot solve the subproblems independently. It stores the answers of subproblems to use them for similar problems.

Scanned with CamScanner

Divide and conquer	Dynamic Programming
1) The problem is divided into small subproblems. Finally, all the solutions of sub-problems are collected together to get the solution to the given problem.	1) The problem is divide into sub-problems but it doesnot solve the subproblems independently. It stores the answers of subproblems to use them for similar problems.
2) Subproblems are independent of each other.	2) Subproblems are interdependent.
3) It is recursive.	3) It is non-recursive.
4) It is a top-down approach.	4) It is a bottom-up approach.
5) It is less efficient.	5) It is more efficient.
6) More time consuming as it solves each sub problem independently.	6) Less time consuming as it uses the answer of the previous subproblems.
7) Eg: Used by merge sort, quicksort, etc.	7) Eg: Used by matrix chain multiplication, optimal binary search tree.

Scanned with CamScanner

classmate
Date
Page

|||| 3) How can dynamic programming technique be applied to the Travelling Salesman Problem?

or

(TSP) - mostly asked in Short Notes

Ans: TSP stands for Travelling Salesman Problem.

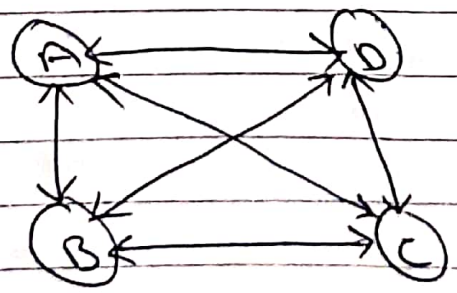
-> It is a process where a salesman has to travel through multiple cities. Here he/she starts from a city, visits all the city and have to return to the initial starting city.

-> This problem uses dynamic programming method to find a ~~path~~ path with lowest path cost.

-> It is also known as hamilton cycle, i.e start from some starting vertex, visit all vertex exactly once and return back to the starting vertex.

Example:

Let, a graph G is given. vertex A, B, C & D denotes 4 cities



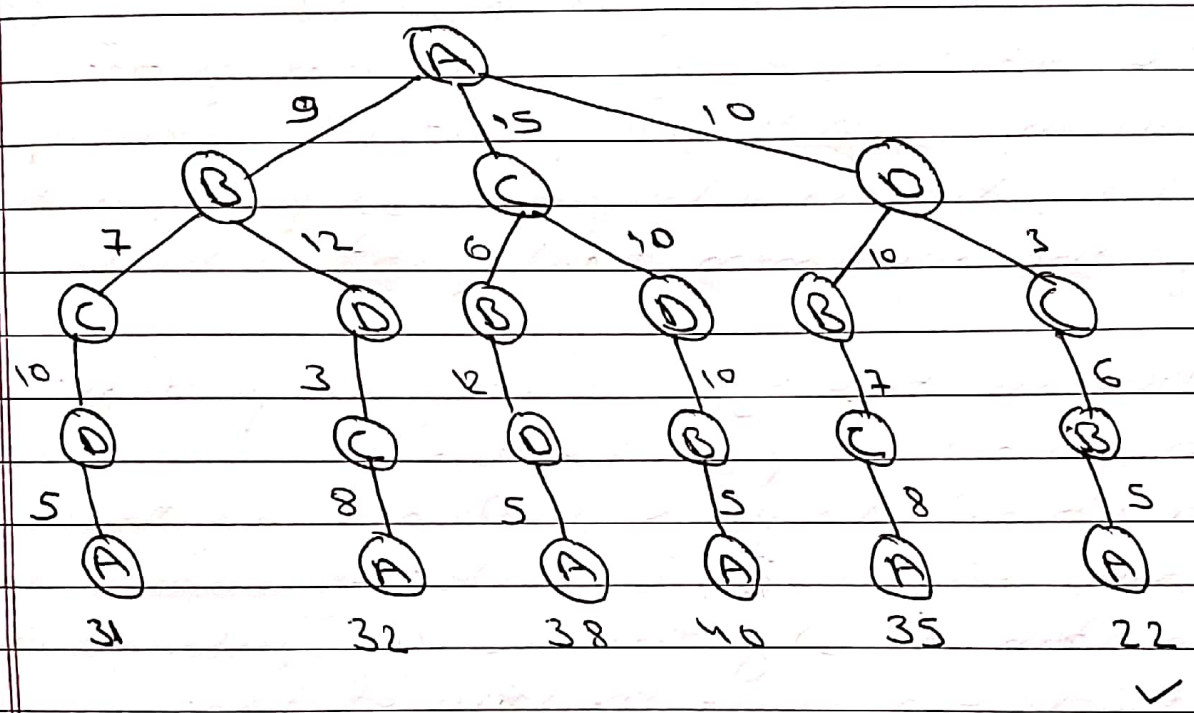
Path cost

	A	B	C	D
A	0	9	15	10
B	5	0	7	12
C	8	6	0	10
D	5	10	3	0

Scanned with CamScanner

To find the optimal solution, we use dynamic programming to find all the possible paths.

Let, start node is A



Here, the shortest path is A, D, C, B, A with path cost 22.

Optional (formula) → Not necessary

$$g(i, S) = \min_{k \in S} \{ c_{ik} + g(k, S - \{k\}) \}$$

Scanned with CamScanner

4) Knapsack Problem

→ Knapsack is like a container or a bag. Suppose we have given some items which have some weights or profits. We have to put some items in the knapsack in such a way ~~the~~ the total value produces a max profit.

→ (Given weights and profit of n items, put these items in a knapsack of capacity W to get the maximum total profit in the knapsack.)

There are two types of knapsack problem.

i) Fractional Knapsack Problem

→ Fractional knapsack problem is for those items which are divisible. Eg. If we have an item of 3kg then we can pick the item of 2 kg and leave the item of 1kg.

→ It is solved by the greedy approach

ii) 0/1 Knapsack Problem

→ 0/1 knapsack problem is for those objects which are indivisible i.e they cannot be divided.

→ In this, the items are either completely or no items are filled in a knapsack.

→ It is solved by dynamic programming

→ Eg. If we have two items with weight 3 and 5 kg respectively. We have a knapsack of 4 kg. Then we can pick first item 3kg. Knapsack still have 1kg of space but we cannot pick 1 kg from 5 kg.

Scanned with CamScanner

So we do not pick 5 kg item.

15) Explain how greedy approach is useful to find the solution to fractional knapsack problem.
(2014 · Spring. 4.a)

Let an example.
Consider 4 items along with their respective weights and profit values. The knapsack has capacity W = 30
Find an optimal solution.

weights w = 5, 10, 20, 30
Profit p = 30, 20, 100, 90

soln :-

Objects (o)	1	2	3	4
Profits (P)	30	20	100	90
Weight (w)	5	10	20	30
Profit/weight (P/w)	6	2	5	3
x	1	0	1	$\frac{5}{30}$

W = 30

We created a table consisting of objects with their weights & profits.
We also calculated the profit/weight ratio.

We will start placing items in knapsack which has highest $\frac{\text{profit}}{\text{weight}}$ ratio

Now,
Item 1 has highest ratio so placing it in a knapsack.
Remaining weight = 30 - 5 = 25

Scanned with CamScanner

Placing item 3 in knapsack since it has the highest ratio.

Remaining weight = 25-20 = 5

Placing item 4 in knapsack since it has the highest ratio

Here, since the remaining weight is 5. We will place 5 unit of item i.e $\frac{5}{30}$

Now the knapsack is full

~~Tota~~ Calculating Total Weight

Total weight = $\sum_{i} w_i$

$w_i \rightarrow$ weight

= 5 + 20 + 5 = 30

Total Profit = $\sum_{i} P_i$

~~= P Item1 P Item 3 + 5 P Item 4~~

= Profit of Item1 + Profit of Item3 + Profit of $\frac{5}{30}$ Item4

$= 30 + 100 + \frac{5}{30} \times 3$

= 130 + 15

= 145

Scanned with CamScanner

In greedy approach

constraint

$\boxed{\sum x_i w_i \leq m}$

Objective

$\boxed{\max \sum x_i p_i}$

6) 0/1 knapsack Problem

or

Discuss the 0/1 knapsack problem and how this problem can be solved? Explain the algorithm.

or, (2017.S.4.b)

(Question will be given) (2015.S.5a)(2016.S.5.a)

Ans: Explain from Question No.4

P.T.O →

Scanned with CamScanner

Example Question:

Objects x_1 x_2 x_3 x_4

weights = {3, 4, 6, 5}

Profits = {2, 3, 1, 4}

Max weight, W = 8

items, n = 4

Creating table,

w →

	i ↓		0	1	2	3	4	5	6	7	8
P_i	w_i	0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2	2	2	2	2	2
3	4	2	0	0	0	2	3	3	3	5	5
4	5	3	0	0	0	2	3	4	4	5	6
1	6	4	0	0	0	2	3	4	4	5	6

(Formula:

$$V[i,w] = \max\{V[i-1,w], V[i-1, w-w[i]] + P[i]\}$$)

→ Here, w_i is placed in ascending order

→ Since W=8, so we have divided weight in 8 cells.

→ ~~The final~~

→ The final weight is received in last cell.

→ In row 1, The weight is 3 so we can only place it in column with weight 3. before it we cannot pick

Scanned with CamScanner

it so all profit is 0. only from that cell profit will be 2.

Similarly, we will ~~solve all the~~ fill all the table

By solving we found out the value of last cell i.e 6

Now checking items whose combination of weight was 6. Searching those items.

x_1	x_2	x_3	x_4
1	0	0	1

We found out that object 1 and 4 is taken in the bag to get the maximum profit.

Total weight in bag = 5+3 = 8

Total Profit = 4+2 = 6

Hence, by using 0/1 knapsack problem objects 1 and 4 is kept in bag to get maximum profit of 6.

Scanned with CamScanner

7) Compare and contrast 0-1 knapsack problem and fractional knapsack problem in terms of their performances.

8) Explain why greedy algorithm doesnot give optimal solution for 0/1 knapsack problem.

→ Greedy algorithm doesnot give optimal solution for 0/1 knapsack problem because

① greedy approach gives optimal solution only for fractional problem. While applying it in 0/1 knapsack problem it will fail.

② Since, we take either all item or none item in 0/1 knapsack problem. It should be solved using dynamic programming to get the maximum profit.

Scanned with CamScanner

Q8) Write the algorithm and define how multistage graph problem can be solved using dynamic programming.

or

Problem will be given to solve multistage graph
i) 2015 (Fall) (4a) ii) 2016 (Fall) (3b)

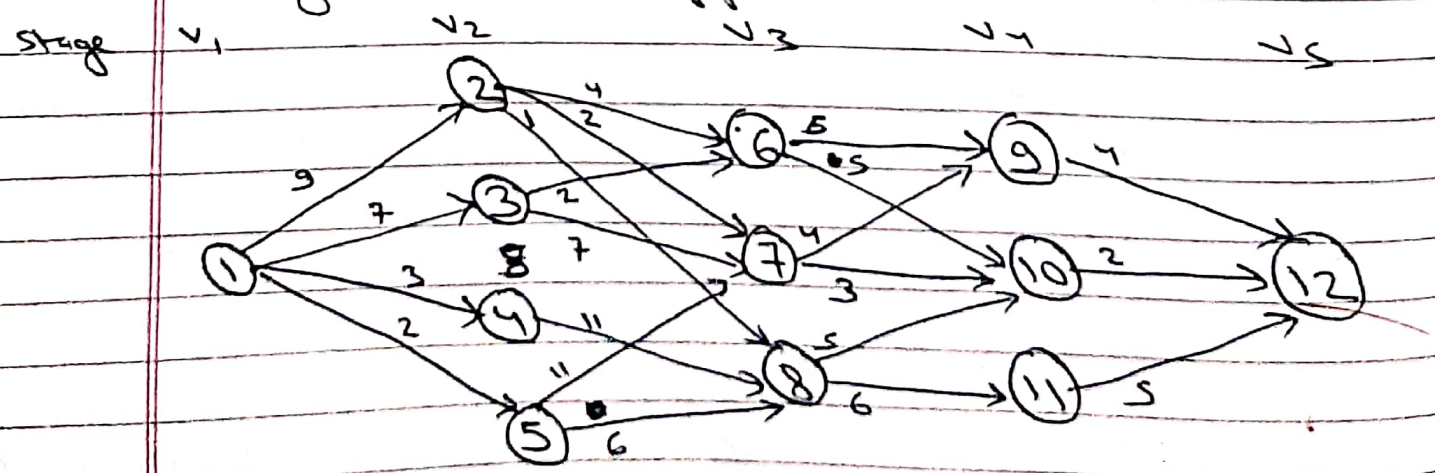
Ans. A multistage is a directed graph in which the nodes can be divided into a set of stages such that all edges are from a stage to next stage only.

-> There is no edge between vertices of same stage and from a vertex of current stage to previous stage.

-> We are given a multistage graph, a source and a destination, we need to find shortest ~~distance from~~ path from source to destination.

Example: 2016 (Fall) 3.b)

Q) Calculate shortest distance and path from 1 to 12 using backward approach



Backward approach

V	1	2	3	4	5	6	7	8	9	10	11	12
Cost	16	7	9	18	15	7	5	7	4	2	5	0
destination	2/3	7	6	8	8	10	10	10	12	12	12	12

Here, the cost of last vertex/destination vertex is 0.
i.e cost of vertex 12 = 0

formula

stage, vertex, cost of edge

$$cost(i,j) = \min_{\substack{\langle j,l\rangle \in E \\ l \in V_{i+1}}} \{ ce(j,l) + cost(i+1, l) \}$$

Now,

$cost(5,12) = 0$

$cost(4,9) = 4$
$cost(4,10) = 2$
$cost(4,11) = 5$

$cost(3,6) = \min\{ ce(6,9) + cost(4,9),$
$ce(6,10) + cost(4,10) \}$
$= \min\{ 6+4, 5+2 \}$
$= 7$

$cost(3,7) = \min\{ ce(7,9) + cost(4,9),$
$ce(7,10) + cost(4,10) \}$
$= \min\{ 4+4, 3+2 \}$
$= 5$

Scanned with CamScanner

$$cost(3,8) = \min\{ce(8,10) + cost(4,10),\ ce(8,11) + cost(4,11)\}$$
$$= \min\{5+2,\ 6+5\}$$
$$= 7$$

$$cost(2,2) = \min\{ce(2,6) + cost(3,6),\ ce(2,7) + cost(3,7),\ ce(2,8) + cost(3,8)\}$$
$$= \min\{4+7,\ 2+5,\ 1+7\}$$
$$= 7$$

$$cost(2,3) = \min\{ce(3,6) + cost(3,6),\ ce(3,7) + cost(3,7\}$$
$$= \min\{2+7,\ 7+5\}$$
$$= 9$$

$$cost(2,4) = \min\{ce(4,8) + cost(3,8)\}$$
$$= 11+7$$
$$= 18$$

$$cost(2,5) = \min\{ce(5,7) + cost(3,7)\ ce(5,8) + cost(3,8)\}$$
$$= \min\{11+5,\ 8+7\}$$
$$= 15$$

$$cost(1,1) = \min\{ce(1,2) + cost(2,2),\ ce(1,3) + cost(2,3),\ ce(1,4) + cost(2,4),\ ce(1,5) + cost(2,5)\}$$

Scanned with CamScanner

$= \min\{9+7, 7+9, 3+18, 2+15\}$
$= \min\{16, 16, 21, 17\}$
$= 16$

Now choosing path.

stage vertex

$d(1,1) =$ vertex 2
$d(2,2) =$ vertex 7
$d(3,7) =$ vertex 10
$d(4,10) =$ vertex 12

Here, the path is 1, 2, 7, 10, 12 with path cost 16.

In this question, the other path is 1, 3, 6, 10, 12 with path cost 16.

Scanned with CamScanner

19) String Editing Problem / Edit Distance

Ans: The string editing problem is to determine the distance between two strings as measured by the minimal cost sequence of deletions, insertions, and changes of symbols needed to transform one string into the other.

We perform three operations to convert one string to another. The operation are

i) Insert (→)
ii) Remove (↓)
iii) Replace (↘)

Conditions

(i) if row ≠ column

Replace ↘	Remove ↓
insert →	

min {(replace, remove, insert) + 1}

(ii) if row == column

Just copy the diagonal

Scanned with CamScanner

Convert string X = a b c f g to Y = a b c e g

Table:-

	null	a	b	c	f	g
null	0	1	2	3	4	5
a	1	0	0+1 → 1	1+1 → 2	2+1 → 3	3+1 → 4
b	2	0+1 → 1	0	0+1 → 1	1+1 → 2	2+1 → 3
c	3	1+1 → 2	0+1 → 1	0	0+1 → 1	1+1 → 2
e	4	2+1 → 3	1+1 → 2	0+1 → 1	0+1 → 1	1+1 → 2
g	5	3+1 → 4	2+1 → 3	1+1 → 2	1+1 → 2	1

← Here, last cell gives the minimum operation. edit ~~distance~~.

∴ Here, to convert string X = a b c f g to string Y = a b c e g we need 1 operation

∴ Minimum cost = 1 unit

Scanned with CamScanner

classmate
Date
Page

1 10) Consider a string editing problem where string X = aabab and Y = babb. Each insertion and deletion has a unit cost and a change cost 2 units. Find a minimum cost edit sequence that performs X in Y.

We know:

Insert (→) Remove (↓) Replace (↘)
(write conditions too)

	null	a	a	b	a	b
null	0	1	2	3	4	5
b	1	0+2 2	1+1 2	2	2+1 3	4
a	2	1	2	2+1 3	2	2+1 3
b	3	1+1 2	1+1 2	2	2+1 3	2
b	4	2+1 3	2+1 3	2	2+1 3	3

Here, insert and remove operation cost 1
replace operation cost 2

So (↘) will be (diagonal element + 2)

Here, the minimum cost edit sequence to perform X in Y is 3 unit.

The operations were
remove one a → 1 unit
replace a to b → 2 unit
Total = 3 unit

Scanned with CamScanner

1) Find the minimum edit distance to transform the string X = aabab into Y = babb

soln:- Here,

We know

Insert (→) Remove (↓) Replace (↘)

Conditions: (write here)

	null	a	a	b	a	b
null	0	1	2	3	4	5
b	1	0+1 1	1+1 2	2	2+1 3	4
a	2	1	1	1+1 2	2	2+1 3
b	3	1+1 2	1+1 2	1	1+1 2	2
b	4	2+1 3	2+1 3	2	1+1 2	2

Here the operations performed are:

a from X is converted to b = 1 unit

a from X is removed = 1 unit

∴ Hence, the minimum cost edit sequence is 2 unit to ~~perform~~ transform string X into Y.

Scanned with CamScanner