15CS204J – Algorithm Design and Analysis

Cycle Test – II Answer key

PART – A

1. B	6. B	11. B	16. C
2. D	7. A	12. D	17. D OR B
3. B	8. D	13. B	18. A
4. A	9. A	14. B	19. B
5. C	10. C	15. C	20. B

PART-B

21)Illustrare the greedy knapsack problem with an example?

Greedy knapsack-Fractional knapsack

Definition

In **Fractional Knapsack**, we can break items for maximizing the total value of knapsack. This problem in which we can break item also called fractional knapsack problem.

The basic idea of greedy approach is to calculate the ratio value/weight for each item and sort the item on basis of this ratio.

Then take the item with highest ratio and add them until we can't add the next item as whole and at the end add the next item as much as we can. Which will always be optimal solution of this problem.

Example

Input:

Items as (value, weight) pairs

arr[] = {{60, 10}, {100, 20}, {120, 30}}

Knapsack Capacity, W = 50

Solution:

Item 1 taken fully

Current weight=10 Total benefit=60

So item 2 can also be taken

Current weight=30 total benefit=160

If item 3 is taken current weight will become 60

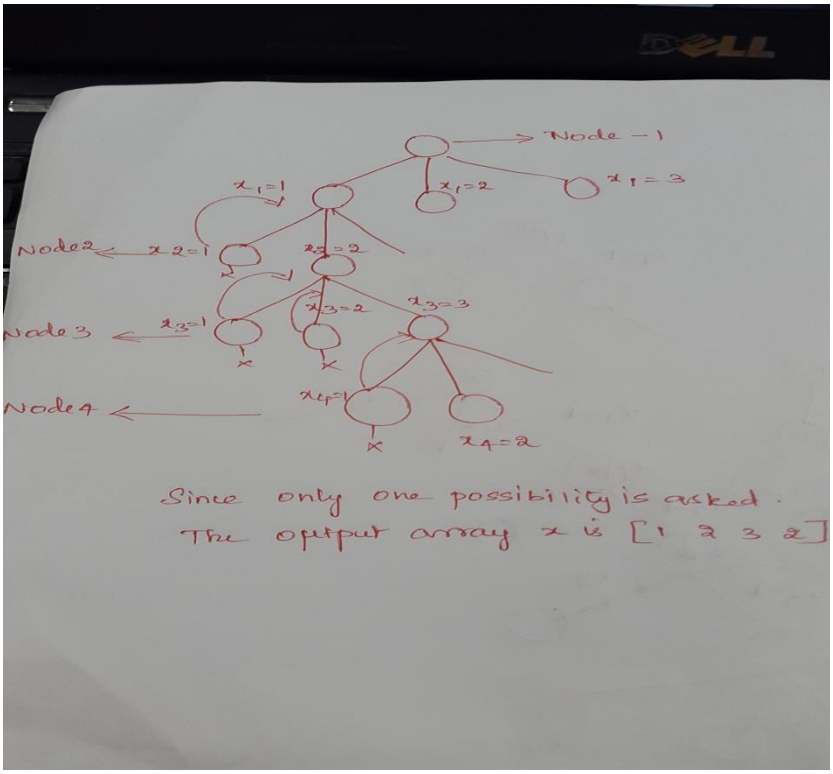
So fraction of item 3 is taken

i.e)(20/30)*120=80

Therfore total benefit=240

22)The minimum number of colour allotted to the graph is 3.If we assign two colours the constraint of adjacency vertices should not be coloured with same colour will not not be met

The state space diagram is



23)

Randomized	**Deterministic**
A **randomized algorithm** is an algorithm that employs a degree of randomness as part of its logic	The deterministic slgorithm does not employ randomness in the logic

The output of the randomized algorithm depends upon the random number selected	A **deterministic algorithm** is an algorithm which, given a particular input, will always produce the same output
Eg:Randomized Quick sort	Eg:Quick sort

24)**NP-Hard Problem:**

A decision problem *H* is NP-hard when for every problem *L* in NP, there is a polynomial-time reduction from *L* to *H*

Example:

Circuit satisfiability problem

The NP-hard problems are handled by reducing the to NP problems

25)Matrix Chain Multiplication:

Given a sequence of matrices, find the most efficient way to multiply these matrices together.

26)Generating Permutation Bruteforce vs Backtracking

Backtracking:

Algorithm:

- Iterate over the string one character at a time.
    - Fix a character at the first position and the use swap to put every character at the first position
    - Make recursive call to rest of the characters.
    - Use swap to revert the string back to its original form for next iteration.

Time Complexity: O (n*n!)

27)General backtracking method

Pick a starting point.

while(Problem is not solved)

For each path from the starting point.

check if selected path is safe,

if yes select it and make recursive call

to rest of the problem

If recursive calls returns true,

return true.

else

undo the current move

return false.

End For

If none of the move works out, return false

PART C

Refer notes