1. branch and bound
2. back tracking
3. Branch vs back track
4. NP, NP Hard, NP Complete
5. Greedy
6. Dynamic Programming
7. One que on **master theorem -** T (n) = T (n/2) + 2^n
8. Recurrence relation
9. omega notation/theta/big oh
10. find a failure function for a particular KMP problem
11. string matching algorithms
12. last occurrence for MINIMIZE
13. matrix multiplication

ANSWERS

1. Branch and Bound is an algorithm that is generally used for solving optimization (mainly minimization) problems which require combinations. A bound is set for each node in the state space tree and while traversing, the node with optimal bound is selected and traversed further. B&B has more pruning compared to backtracking algo.

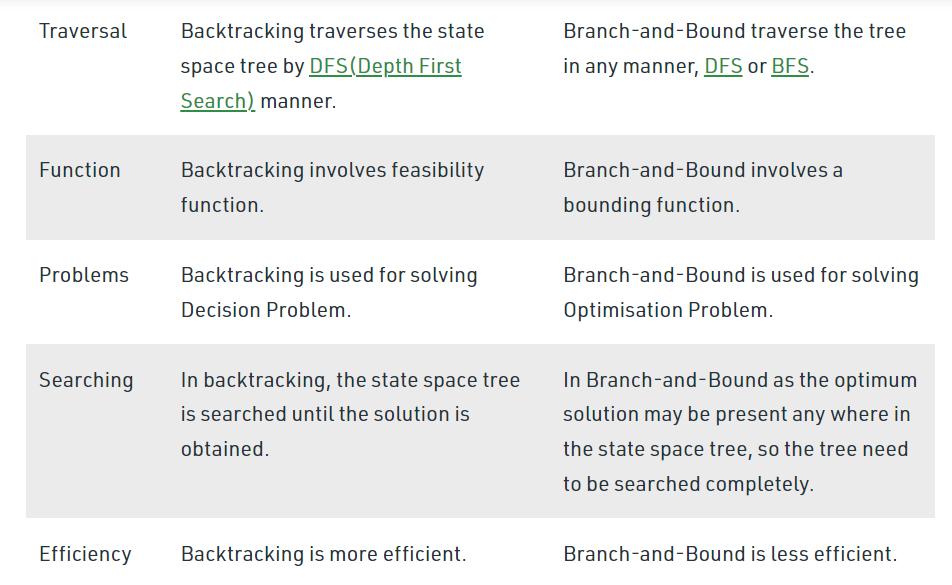
2. Backtracking can be defined as a general algorithmic technique that considers searching every possible combination in order to solve a computational problem.  solving all sub-problems one by one in order to reach the best possible solution.

3. first tell the definitions from above



Traversal BT uses DFS to traverse through B&B uses BFS to traverse.

Sst



4. NP complete – An NP complete problem is one which is both NP hard and has a non-deterministic polynomial solution to it.

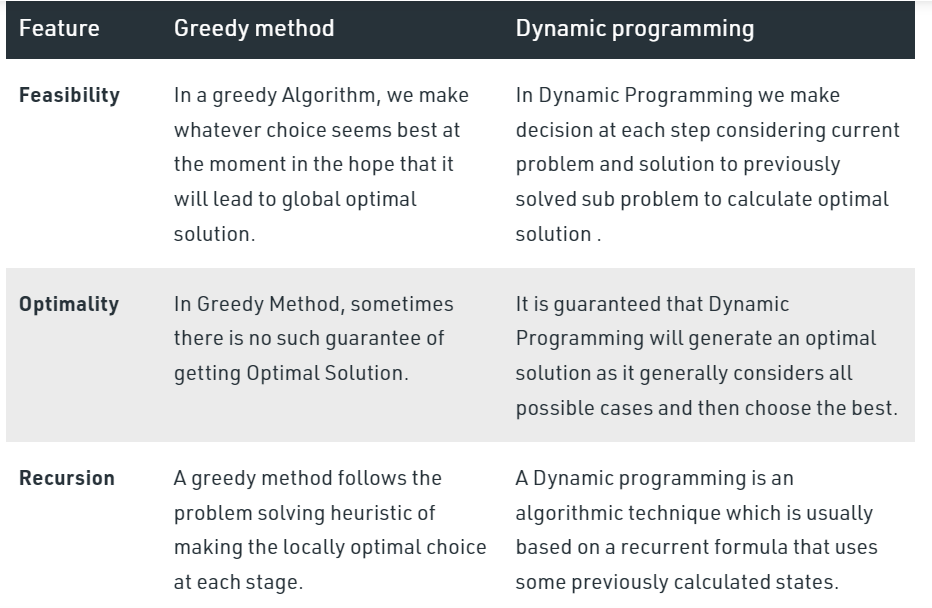
Eg- travelling salesman problem,

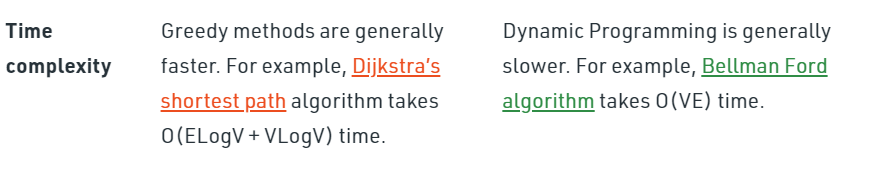
NP Hard – A problem is called NP Hard if there exists an NP complete problem which is polynomially reducible to it. a problem is NP-hard if it is difficult to solve, or find a solution.

NP – It is the set of problems which are non-deterministic and can be solved in polynomial time.  a problem is in NP if it is easy to verify its solutions.

5. Greedy algorithms are mainly used for optimization problems, this algorithm takes the optimal choice at each step to eventually arrive at the overall optimum and find the solution.

6. Dynamic programming has an very important feature that is memoization, where the results of the subproblems or the previous computations are stored and it is used for further computations, this helps to greatly reduce the running time.

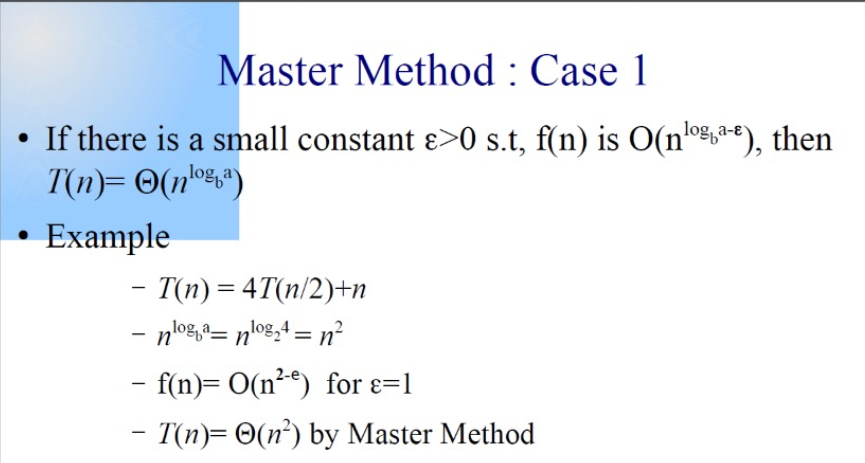


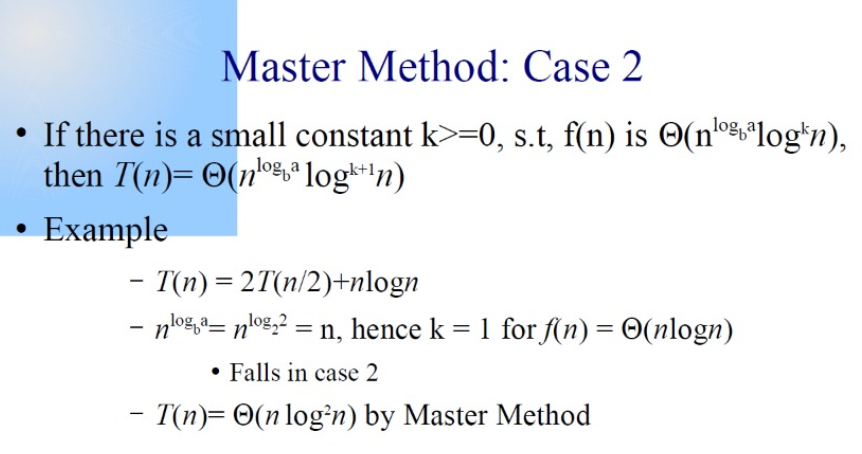


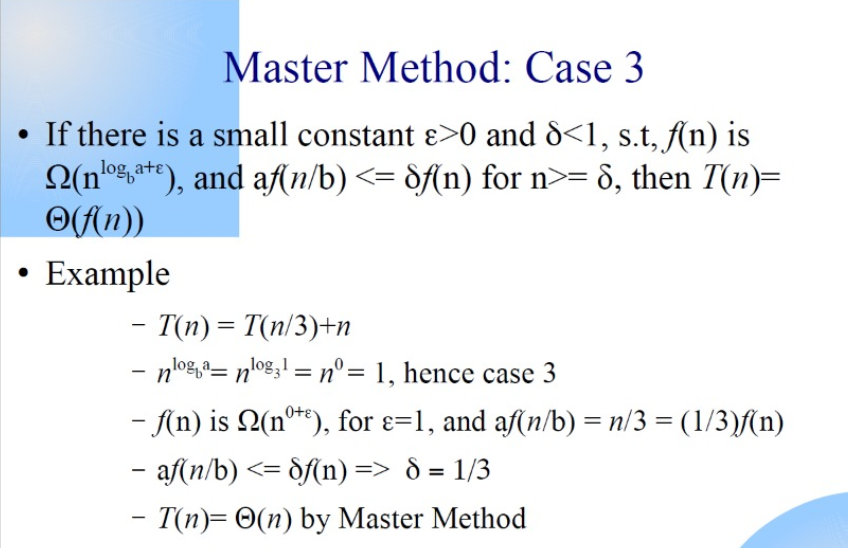
7. a – the number of times the sub problem is divided into(size of subproblem in recursion)

b – the factor by which the sub problem is divided each time(size of each subproblem)

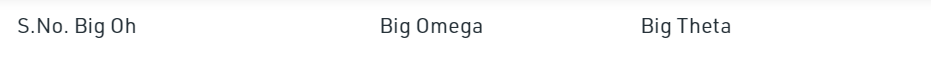
n – total problem size

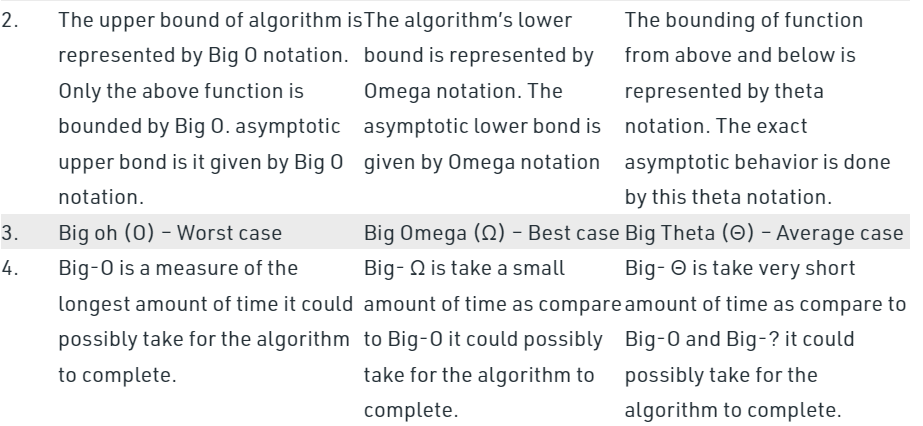






8. A recurrence relation is an equation that defines a sequence based on a rule that gives the next term as a function of the previous term(s).

9. 



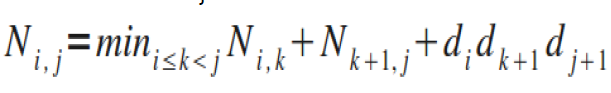
10.

11. rabin karp – uses hash function, it will create a hash function for both the patten and the substrings of the string when the hash function values match, the pattern and the substring will be checked for match.

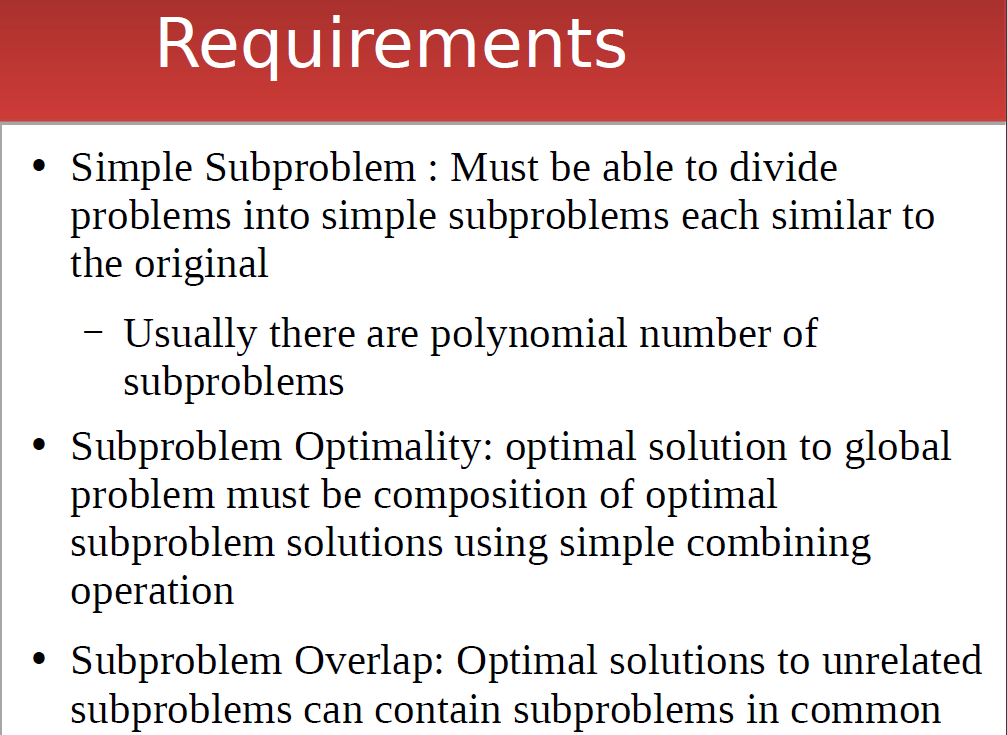
Boyer moore – bad character heuristic, using a function (max(1,len(pattern-index(char)-1))) it will assign it to each character in the pattern. The pattern will start matching with the string from the end and if mismatch occurs, it will shift the number of characters assigned to the pattern using the above mentioned function.

12. find the unique characters and find the last occurrence of them.

13.



Requirements for DP:



Formula for LCS:

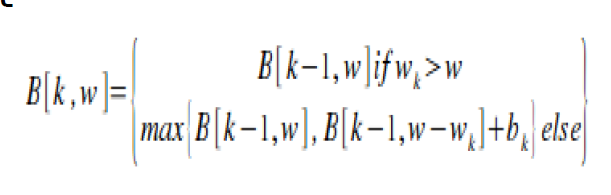
if (A[i] == B[j])

L[i,j] = L[i-1,j-1]+1

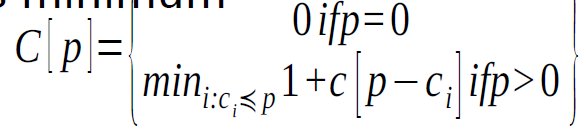
else

L[i,j] = max{L[i-1,j],L[i,j-1]}

Formula for 0-1 knap sack:



Formula for Coin Change prb:



Greedy Algo:

Fractional Knapsack - O(n log n)

Job Sequencing - O(n^2) or O(n log n) - if max heap

Huffman code - O(n log n) - If there are n nodes, extractMin() is called 2\*(n – 1) times. extractMin() takes O(logn) time as it calles minHeapify().

Kruskal - O(n log n) - O(E log E) - Min cost spanning tree

Prims - O(V^2) - adj matrix -- O(E log V) - adj list - Min cost spanning tree

Dijkstra - O(V^2) or O(E log V) -adjacency list with binary heap.

Dynamic prog

Multi stage graph - O(n^2)

Floyd Warshall - All pair shortest path - O(V^3)

Bellman Ford - O(EV)

0/1 Knapsack - O(N\*W) - N is the number of weight element and W is capacity

Matrix chain mul - O(n^3)

Longest sub sequence - O(nm)

Branch and Bound - Optimisation problems - minimisation

FIFO BB - Queue

LIFO BB - Stack

Least cost BB

Rabin-Karp - O(nm) - hashing

KMP - O(nm)