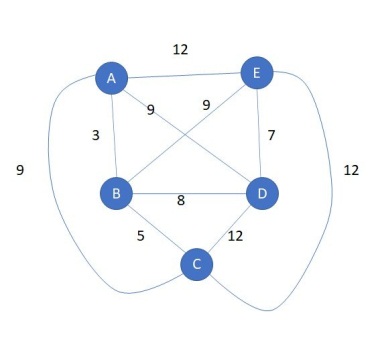
SET- B Key

1. In what manner is a state-space tree for a backtracking algorithm constructed?
   1. Breath First Search
   2. Depth First Search
   3. Nearest Neighbour method
   4. Twice around the tree
2. For how many queens was the extended version of Eight Queen Puzzle applicable for n\*n squares?
   1. 5
   2. 8
   3. 6
   4. n
3. What is the Time Complexity of DFS?
   1. O(V + E)
   2. O(E)
   3. O(V)
   4. O(V \* E)
4. What is the optimal solution for the Travelling Salesman Problem?



* 1. A-B-D-E-C-A
  2. A-B-C-E-D-A
  3. A-B-C-D-E-A
  4. A-D-E-B-C-A

1. Which is true statement.
   1. Breadth first search is shortest path algorithm that works on un-weighted graphs
   2. Depth first search is shortest path algorithm that works on un-weighted graphs.
   3. Both of above are true.
   4. None of above are true.
2. Which of the following options match the given statement:

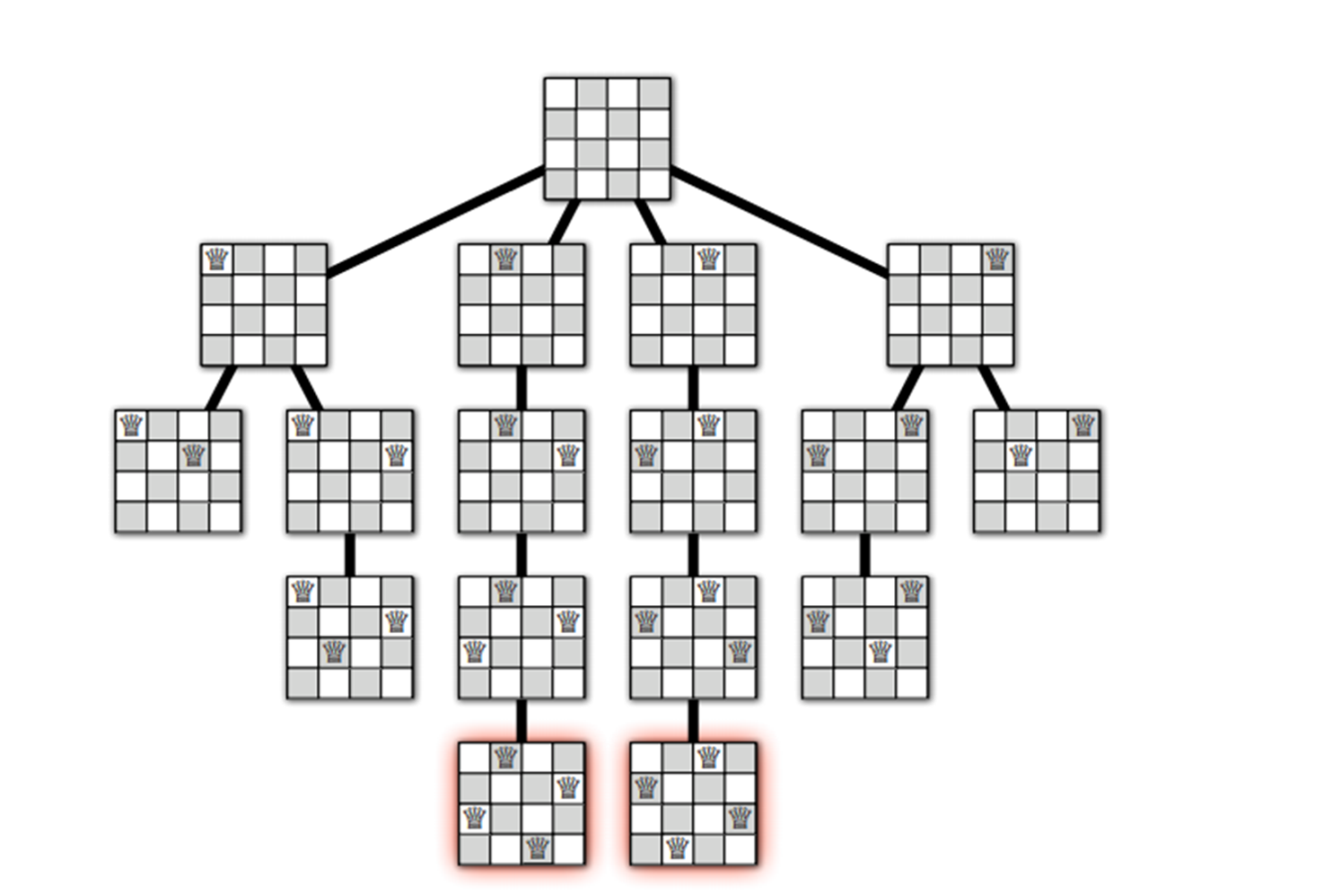
Statement: The algorithms that use the random input to reduce the expected running time or memory usage, but always terminate with a correct result in a bounded amount of time.

1. Las Vegas Algorithm
2. Monte Carlo Algorithm
3. Atlantic City Algorithm
4. None of the mentioned
5. What is a Rabin and Karp Algorithm?  
   a) String Matching Algorithm  
   b) Shortest Path Algorithm  
   c) Minimum spanning tree Algorithm  
   d) Approximation Algorithm
6. Problems that can be solved in polynomial time are known as?
7. Intractable
8. **Tractable**
9. Decision
10. Complete
11. Choose the correct answer for the following statements:
12. The theory of NP–completeness provides a method of obtaining a polynomial time

for NP algorithms.

1. All NP-complete problem are NP-Hard
   1. I is FALSE and II is TRUE
   2. I is TRUE and II is FALSE
   3. Both are TRUE
   4. Both are FALSE
2. Which of the following is true
   1. NP is subset of P
   2. P is subset of NP
   3. P and NP are equal
   4. None of the above

Q11. N Queen’s problem for 4\*4 chess board



Q 12. Travelling Salesman Problem

**12**

**11**

**3**

**6**

**7**

**18**

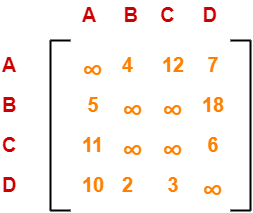
**2**

**10**

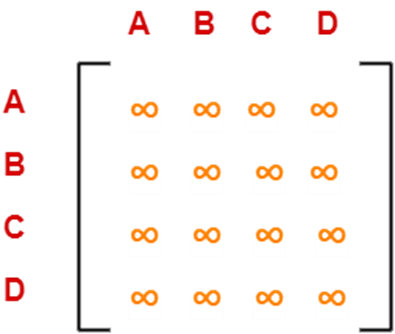
**5**

**4**

Ans:



……..After reduction



Cost(7)

= cost(6) + Sum of reduction elements + M[D,B]

= 25 + 0 + 0

= 25

Thus,

• Optimal path is: **A → C → D → B → A**

• Cost of Optimal path = **25 units**

**Q13.Depth first Search with example**

* Depth first search (DFS) algorithm starts with the initial node of the graph G, and then **goes to deeper and deeper until we find the goal node** or **the node which has no children.**
* It is an algorithm **for traversing the tree or graph data structure**
* Application of DFS – Topological Sorting, Finding the bridges of a graph, Cycle Detecting

Algorithm:

DFS-iterative (G, s): //Where G is graph and s is source vertex

let S be stack

S.push( s ) //Inserting s in stack

mark s as visited.

while ( S is not empty):

//Pop a vertex from stack to visit next

v = S.top( )

S.pop( )

//Push all the neighbours of v in stack that are not visited

for all neighbours w of v in Graph G :

if w is not visited :

S.push( w )

mark w as visited

DFS-recursive(G, s):

mark s as visited

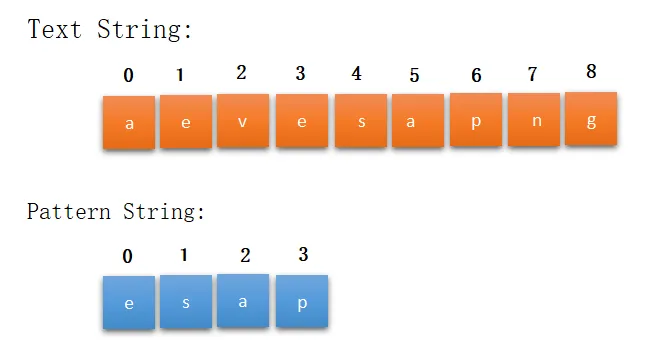
for all neighbours w of s in Graph G:

if w is not visited:

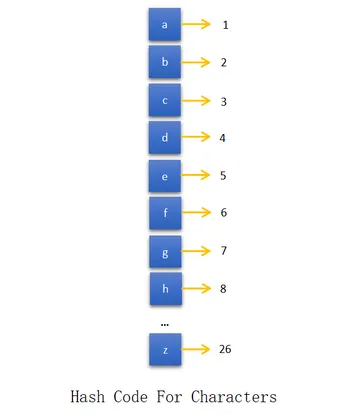
DFS-recursive(G, w)

* **Example with step by step process**

[**Q14.**](https://www.tutorialcup.com/interview/string/rabin-karp-algorithm.htm) **Rabin karp String Matching**

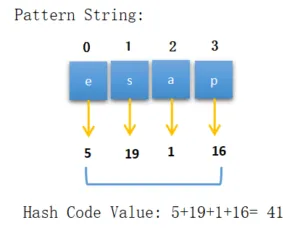


Hash code for the characters is listed below.



**Step-1:**

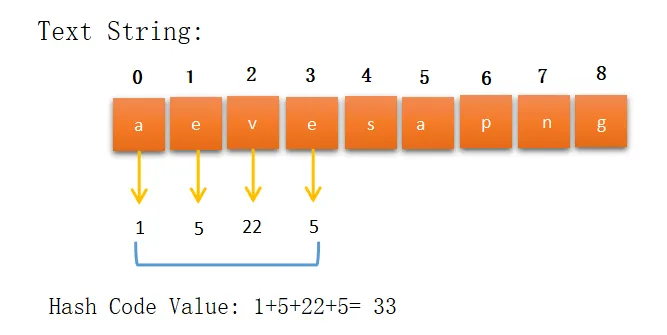
Find the hash code value of pattern string using the hash code assigned for characters.



Hash code = 41mod 9

= 5

Step 2:

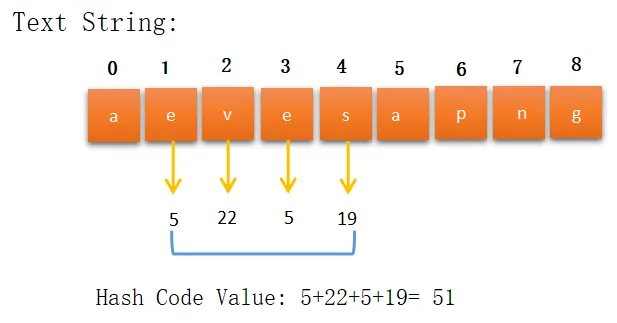


Hash code = 33 mod 9

= 6

Hash code value is not the same then we move to the next substring of length M(4).

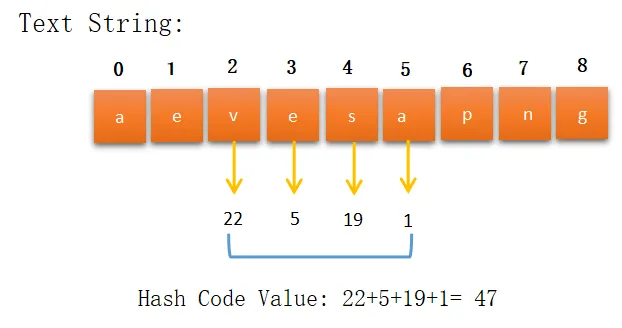
Step 3:



Hash code= 51 mod 9

= 6

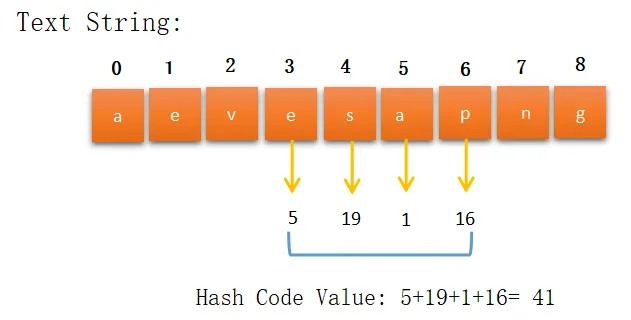
Step 4:



Hash code= 47 mod 9

= 2

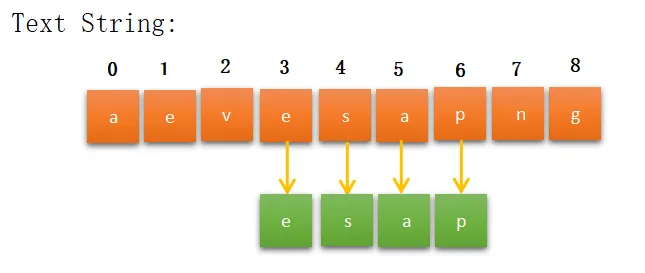
Step 5:



**Hash code = 41 mod 9**

**= 5**

Hash code value is the same here, so we check the substring characters one by one with the pattern string.



All the characters matched then we print the starting index of the substring and move to the next substring of length M if possible.

**Q15. NP Complete with example**

NP Complete:

* The name "NP-complete" is short for "nondeterministic polynomial-time complete"
* It is a problem for which the correctness of each solution can be verified quickly (namely, in polynomial time) and a brute-force search algorithm can find a solution by trying all possible solutions.
* The problem can be used to simulate every other problem for which we can verify quickly that a solution is correct. In this sense, NP-complete problems are the hardest of the problems to which solutions can be verified quickly. If we could find solutions of some NP-complete problem quickly, we could quickly find the solutions of every other problem to which a given solution can be easily verified.

Example:

**Hamiltonian Cycle Problem**

**Definition:** A Hamiltonian cycle is a cycle in a graph that visits each vertex exactly once.

To show Hamiltonian Cycle Problem is NP-complete, we first need to show that it actually belongs to the class NP, and then use a known NP-complete problem to Hamiltonian Cycle.

Does Hamiltonian Cycle Problem ∈ NP?

**Given:** 𝐺𝑟𝑎𝑝ℎ 𝐺 = (𝑉, 𝐸)

**Certificate:** List of vertices on Hamiltonian Cycle

To check if this list is actually a solution to the Hamiltonian cycle problem, one counts the vertices to make sure they are all there, then checks that each is connected to the next by an edge, and that the last is connected to the first.

It takes time proportional to n, because there are n vertices to count and n edges to check. n is a

polynomial, so the check runs in polynomial time.

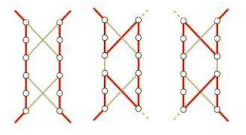
Therefore, Hamiltonian Cycle ∈ NP.

Prove Hamiltonian Cycle Problem ∈ NP-Complete

**Reduction:** Vertex Cover to Hamiltonian Cycle

**Definition:** Vertex cover is set of vertices that touches all edges in the graph.

Given a 𝑔𝑟𝑎𝑝ℎ 𝐺 and integer k, construct a 𝑔𝑟𝑎𝑝ℎ 𝐺’ such that 𝐺 has a vertex cover of size k iff 𝐺’ has a 𝐻𝑎𝑚𝑖𝑙𝑡𝑜𝑛𝑖𝑎𝑛 𝑐𝑦𝑐𝑙𝑒.

**Idea**: To construct widget for each edge in the graph.

As shown above, there are three ways to traverse a widget;

1. Enter from u, go somewhere else in the graph, and then come back through the other side i.e v

2. Enter and Exit through u

3. Enter and Exit through v

Construct 𝐺’ for 𝐺 (Vertex cover) of size 𝑘 = 2

With the construction, any graph with a vertex cover, can be used to make a graph with a Hamiltonian Cycle graph. Since creating such a graph can be done under polynomial time, simply replace edges with widgets and make proper connections, we have a reduction from Vertex Cover to Hamiltonian Cycle.

This means that finding whether a graph has a Hamiltonian Cycle or not is NP Hard. As we have seen earlier it’s also in NP, therefore, Hamiltonian Cycle is an NP Complete Problem