LIKHIT JHA:

LQ1

How many bishops can you set a **6 \* 6** chess boards ?

Use this board and try it out for yourself

Hint:A bishop can travel in any of the four diagonals.

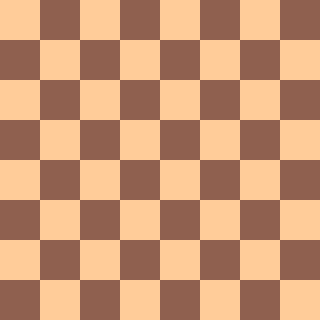
Options are:

8

10

6

12



SOLUTION

2\*(n-1)

=2\*(n-1)

where n is the size of the chessboard

-Place n bishops in the first row

-Place n-2 bishops in the last row. We only leave two corners of last row

Likhit Jha: Question 2

Is the following code a :

1) 0/1 Knapsack

2) Floyd's Warshall Algorithm

3) N-Queens problem

4) Longest Common Subsequence problem

#define max 100

int N=10;

bool CheckIF(int graph[max][max], int x, int y) {

for (int i = 0; i < y; i++)

if (graph[x][i])

return false;

for (int i=x, j=y; i>=0 && j>=0; i--, j--)

if (graph[i][j])

return false;

for (int i=x, j=y; j>=0 && i<N; i++, j--)

if (graph[i][j])

return false;

return true;

}

bool Solve(int graph[max][max], int y) {

if (y == N) {

printGraph(graph);

return true;

}

for (int i = 0; i < N; i++) {

if (CheckIF(graph, i, y) ) {

graph[i][y] = 1;

if ( Solve(graph, y + 1))

return true;

graph[i][y] = 0; }

}

return false;

}

void checkSolution() {

int graph[max][max];

memset(graph, 0, sizeof(graph));

if ( Solve(graph, 0) == false ) {

cout << "Solution does not exist";

return ;

}

return ;

}

int main() {

checkSolution();

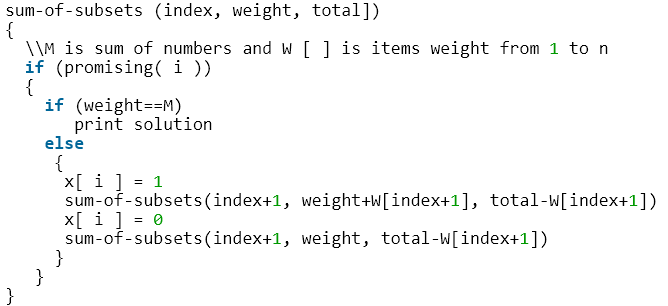
return 0;

}

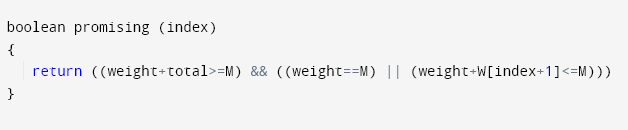
SOLUTION LIKHIT JHA

N-Queens problem

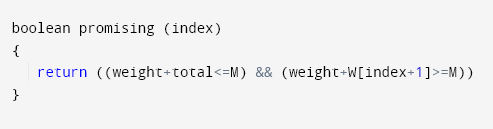
Consider the following code and choose the appropriate option



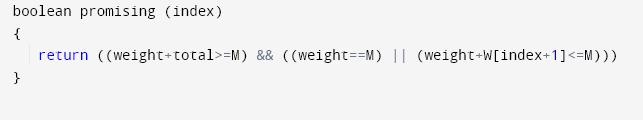
1)



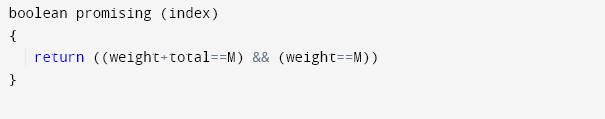
2)



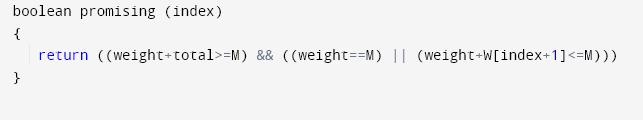
3)



4)



Solution



Option 3

promising(i): checks if the partial solution represented by I, can lead to the required solution

weight: weightSoFar

total: totalPossibleLeft

M: Given total sum needed to be formed using subsets

Which of the following is not **false** with respect to back tracking ?

1. Backtracking enumerates a list of Non-promising nodes that could be computed to give the possible solutions of a given problem.
2. N queen problems cannot be solved using backtracking
3. Bruteforce is faster than backtracking
4. Backtracking algorithm is implemented by constructing a tree of choices called as State-space tree

Solution :

4) Backtracking algorithm is implemented by constructing a tree of choices called as State-space tree

**Vasav Patel:**

**Question-1**

Choose the correct option related to branch and bound algorithms.

A)Its complexity is O(2n).

B)It does not generate nodes according to several rules.

C)It partially searches the space tree to achieve an optimal solution and it demands a feasibility solution.

D)It brings to end nodes as soon as it guarantees that no solution to the problem can be gained by choices which correlate with nodes' descendants.

**Answer:-D)**It brings to end nodes as soon as it guarantees that no solution to the problem can be gained by choices which correlate with nodes' descendants.

**Question-2**

If the complete graph has 703 edges,Find out how many Hamilton cycles can be obtained?

A)25!

B)27!

C)37!

D)31!

**Answer:-C)**It is the complete graph.The number of edges of the graph is 703 with the help of this equation(N-1)! We can find out Hamilton cycles.

First we have to find out vertices of the graph,We have given 703 edges which means edges=N(N-1)/2=703 so the N(N-1)=1406 Now square root of 1406=37.49666 round up to 38.So we get N=38 vertices. With the help of vertices we can get a Hamilton cycle. Hamilton equation (N-1)! =(38-1)!= 37! Cycle we can obtain.

**Question-3**

Alex is planning a trip with his friend to tour all over the USA. First of all, they fly out of New Orleans, then they visit Washington DC, after that they visit LA and end up at Alex's old house in New Jersey. For this scenario which graph would be best?

A)Hamiltonian circuit

B)Hamiltonian path

C)Euler path

D)Euler circuit

**Answer:B)**Hamiltonian path

**Question-4**

Tyler has to keep checking up stop signs at each crossing in his hometown Philadelphia. He parks his Jeep in the transportation department parking lot and then he needs to end up back at his jeep to go home. He only wants to check each crossing one time. Which graph would be best for this scenario?

A)Hamiltonian circuit

B)Hamiltonian path

C)Euler path

D)Euler circuit

**Answer: A)**Hamiltonian circuit/cycle

**Question-5**

Select Option which is more suitable related to branch and bound algorithms.

A)It is applied to solve minimization problems, and maximization problems.

B) It completely searches the state space tree to get an optimal solution and it involves a bounding function.

C)It finds the solution to the overall issue by finding a solution to the first subproblem and then recursively solving other subproblems based on the solution of the first issue.

D)it is not a sophisticated method and It is more functional than backtracking.

**Answer:- B)**(It completely searches the state space tree to get an optimal solution and it involves a bounding function.)

Questions:bhavika

Unit4

Q1. Which of the following are not true?

a. Travelling salesperson problem can be solved only by using either dynamic programming or brute force method.

b. Dijkstra’s algorithm is the fastest way to solve single source shortest path problems.

c. Overlapping subproblems cannot be handled by the greedy approach.

d. Floyd Warshall's algorithm runs for ‘n’ iterations for ‘n’ nodes.

**Ans: a**

**It can be solved by using 3 methods-dynamic programming,brute force method and branch and bound method also.**

Q2 . A carton is to be filled using the 0/1 Knapsack algorithm. Maximum items available are 4. Maximum weight that can be held by the carton is 7. Item 1

Weighs 1,item 2 weighs 3,item 3 weighs 4,item 4 weighs 5.Their corresponding value are Rs. 1,4,5,7. Following is maximum value after filling the carton to optimum quantity:

1. 12
2. 9
3. 8
4. 11

Ans: b

Items of weight 3,4 selected.

Q3 . Length of longest common subsequence of ‘PQRTSM’ and ‘LQSTM’ is:

a. 2

b. 3

c. 4

d. 5

**Ans: 3(b) lcs=**

Q4. For a given problem, time complexity of dynamic programming is not \_\_\_\_ than the greedy approach.

a. Higher

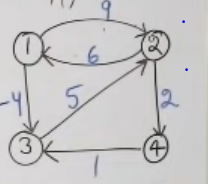
b. Lower

C. can’t determine

D. different

**Ans:b dynamic programming is generally slower as it works in iterations and looks back at previous decisions.**

Q5. Using Floyd’s Algorithm : D2[i=2][j=3] for following graph is (i=1,2,3,4 and j=0,1,2,3,4 ) :



a. 2

b. Infinity

c. 5

d. 9

**Ans: a**

Q6 . Path length for a travelling salesperson problem with n nodes is :

a. Path length=n

b. Path length=n+1

c. path length<=n

d. path length>=n

**Ans: b**

**Salesperson starts from source node,travels rest of the n-1 nodes and returns back to the source node. So 1+(n-1)+1=n+1**

Ritu Pujari

Unit 6

1. Assuming a set of shopping malls and the distance between each pair of malls as an adjacency matrix, the problem is to visit each shopping mall exactly once and find the shortest possible path back to the starting point. Identify if the given problem is:
2. NP Hard
3. NP Complete
4. P
5. None of the mentioned

Ans: b (The above given question is a travelling salesman problem which belongs to Np-Complete class)

2. Class NP is a collection of all decision problems:

1. That can be solved using polynomial time algorithm
2. That cannot be solved using polynomial-time algorithms.
3. Have polynomial time algorithms to test potential solutions
4. None of the above

Answer: c (NP is the set of all [decision problems](https://en.wikipedia.org/wiki/Decision_problem) for which the answers can be ***verified*** in polynomial time)

3. Which of the following statements are known to be true?

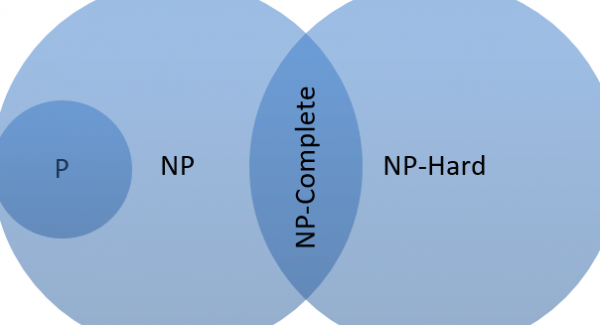
A) P = NP

B) NP ⊆ P

C) P ⊆ NP

D) None of the above

Answer c



4.Which of the following statements about NP-Complete and NP-Hard problems are correct?

(A) None of the above

(B) The first problem that was proved as NP-complete was the circuit satisfiability problem.

(C) P is a subset of NP Hard

(D) All of the above

Answer : b

**Q1 Aryan. . Identify the correct Bellman Ford Algorithm.**

**a)**

**for i=1 to V[g]-1**

**do for each edge (u,v) in E[g]**

**do Relax(u,v,w)**

**for each edge (u,v) in E[g]**

**do if d[v]>d[u]+w(u,v)**

**then return False**

**return True**

**b)**

**for i=1 to V[g]-1**

**for each edge (u,v) in E[g]**

**do if d[v]>d[u]+w(u,v)**

**then return False**

**return True**

**c)**

**for i=1 to V[g]-1**

**do for each edge (u,v) in E[g]**

**do Relax(u,v,w)**

**for each edge (u,v) in E[g]**

**do if d[v]<d[u]+w(u,v)**

**then return true**

**return True**

**d)**

**for i=1 to V[g]-1**

**do for each edge (u,v) in E[g]**

**do Relax(u,v,w)**

**return True**

**Answer: a**

**Q2 Aryan.** Who developed the “ Principle of optimality ” in the concept of dynamic programming?

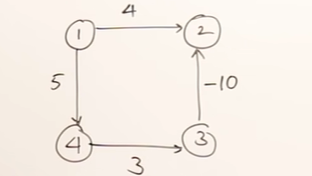
a) Richard bellman

b) Meril M Flood

c) Floyd Warshall

d) George Dantzig

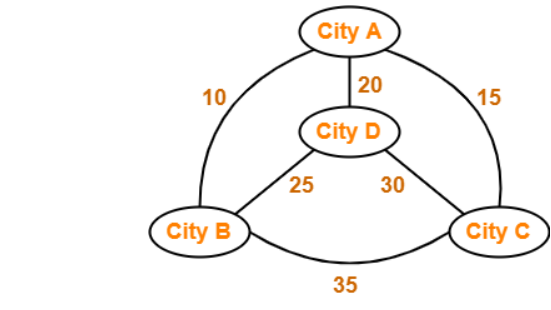
Q3.Find the shortest path for vertex 2 and 4 using bellman ford’s algorithm?



1. -2 , 0
2. 0, -2
3. -2, 5
4. 2, 5

Answer (c)

Q4 Aryan.In the figure given below using the concept of travelling salesman problem find out the shortest possible path to complete the tour starting from A?

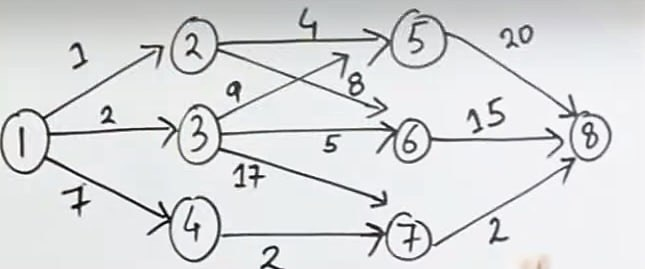


1. 70
2. 80
3. 85
4. 90

Answer:b

Q5 Aryan. Using the concept of “multistage process” refer to the fig given below and find out the minimum cost of fun(2, 2)?

Hint: use recursive equation



1. 24
2. 23
3. 11
4. 25

Answer:a

Q6) Aryan ( EXTRA) Identify the incorrect statement?

1. Bellman-Ford algorithm is used **to find the shortest path from the source vertex to every vertex in a weighted graph**.
2. An algorithm that is used for finding the shortest distance, or path, from starting node to target node in a weighted graph is known as Dijkstra’s Algorithm.
3. The bellman ford algorithm cannot find the shortest distance to every vertex in the weighted graph with the negative edges.
4. In a multistage graph there is no edge between vertices of same stage and from a vertex of current stage to previous stage

QUESTION 6

Is the following code a :

#define max 100

int N=10;

bool CheckIF(int graph[max][max], int x, int y) {

for (int i = 0; i < y; i++)

if (graph[x][i])

return false;

for (int i=x, j=y; i>=0 && j>=0; i--, j--)

if (graph[i][j])

return false;

for (int i=x, j=y; j>=0 && i<N; i++, j--)

if (graph[i][j])

return false;

return true;

}

bool Solve(int graph[max][max], int y) {

if (y == N) {

printGraph(graph);

return true;

}

for (int i = 0; i < N; i++) {

if (CheckIF(graph, i, y) ) {

graph[i][y] = 1;

if ( Solve(graph, y + 1))

return true;

graph[i][y] = 0; }

}

return false;

}

void checkSolution() {

int graph[max][max];

memset(graph, 0, sizeof(graph));

if ( Solve(graph, 0) == false ) {

cout << "Solution does not exist";

return ;

}

return ;

}

int main() {

checkSolution();

return 0;

}

Question 22

**Identify the correct Bellman Ford Algorithm.**

**a)**

**for i=1 to V[g]-1**

**do for each edge (u,v) in E[g]**

**do Relax(u,v,w)**

**for each edge (u,v) in E[g]**

**do if d[v]>d[u]+w(u,v)**

**then return False**

**return True**

**b)**

**for i=1 to V[g]-1**

**for each edge (u,v) in E[g]**

**do if d[v]>d[u]+w(u,v)**

**then return False**

**return True**

**c)**

**for i=1 to V[g]-1**

**do for each edge (u,v) in E[g]**

**do Relax(u,v,w)**

**for each edge (u,v) in E[g]**

**do if d[v]<d[u]+w(u,v)**

**then return true**

**return True**

**d)**

**for i=1 to V[g]-1**

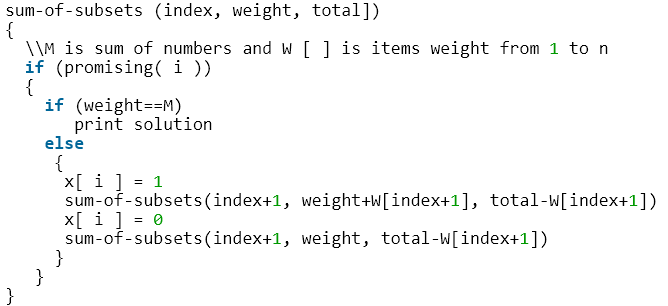
**do for each edge (u,v) in E[g]**

**do Relax(u,v,w)**

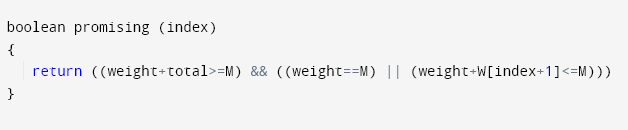
**return True**

**Question 23**

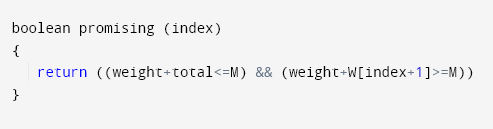
Consider the following code and choose the appropriate option



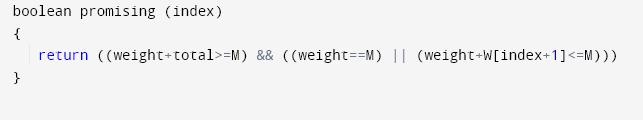
1)



2)



3)



4)

