**Experiment no: 4**

**Backtracking Algorithm**

**Aim:** To implement the following Algorithms using the Backtracking Algorithm

i) N-Queens Problem

ii) Sum of Subset Problem

iii) M-Coloring Problem

iv) Hamiltonian Cycles

v) 0/1 Knapsack problem

**Theory:**

* Backtracking is a systematic approach to problem-solving that involves trying different solutions and undoing choices when they are found to be incorrect.
* It is commonly used for problems that require finding all possible solutions or a specific solution within a large search space.
* The algorithm explores the search space incrementally by building a solution one step at a time and backtracking when it reaches an invalid or unsatisfactory state.
* Recursion is often used to handle the search process, with each recursive call representing a step or decision in the problem-solving process.
* The algorithm maintains a state or configuration that represents the progress made so far in the search.
* At each step, the algorithm makes choices or selects options available and evaluates their validity or desirability.
* Bounding functions are used to determine if a partial solution can possibly lead to a valid solution or if it should be abandoned to save computational resources.
* If a choice leads to a valid solution, it is accepted and the algorithm continues to explore further.
* If a choice leads to an invalid or unsatisfactory state, the algorithm backtracks, undoing the choice and trying an alternative option.
* Backtracking continues until either a solution is found or all possible options have been exhausted.
* The algorithm often utilizes pruning techniques to optimize the search process by eliminating unnecessary or redundant choices.
* Backtracking is applicable to problems with "optimal substructure," meaning that a solution can be constructed from solutions to smaller subproblems.
* It is commonly used in various problem domains such as combinatorial optimization, constraint satisfaction, puzzles, graph problems, and more.

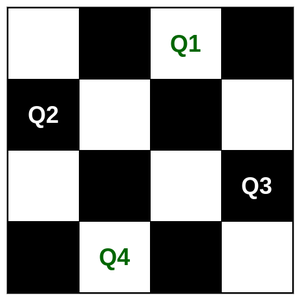
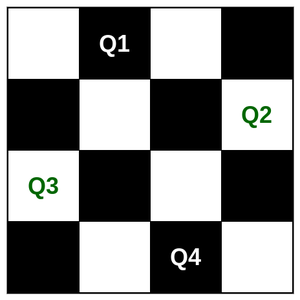
Date:

**i)N-Queens Problem**

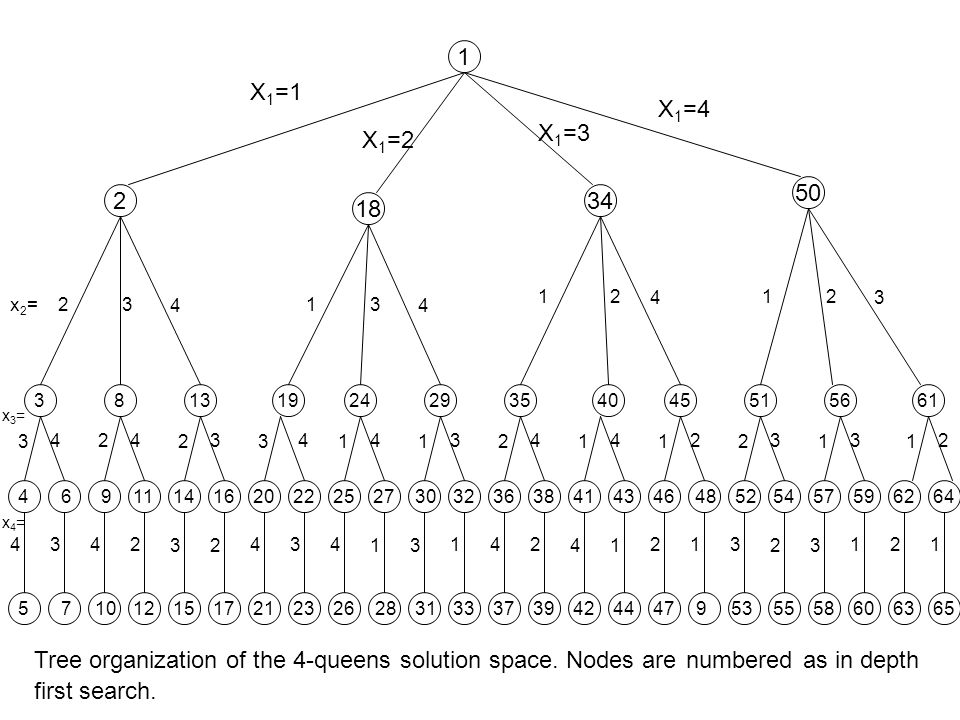
**Problem Statement:**

Write a C program to find all the possible placements of N queens on an NxN board, such that they don't attack each other, thereby solving the N-Queens Problem.

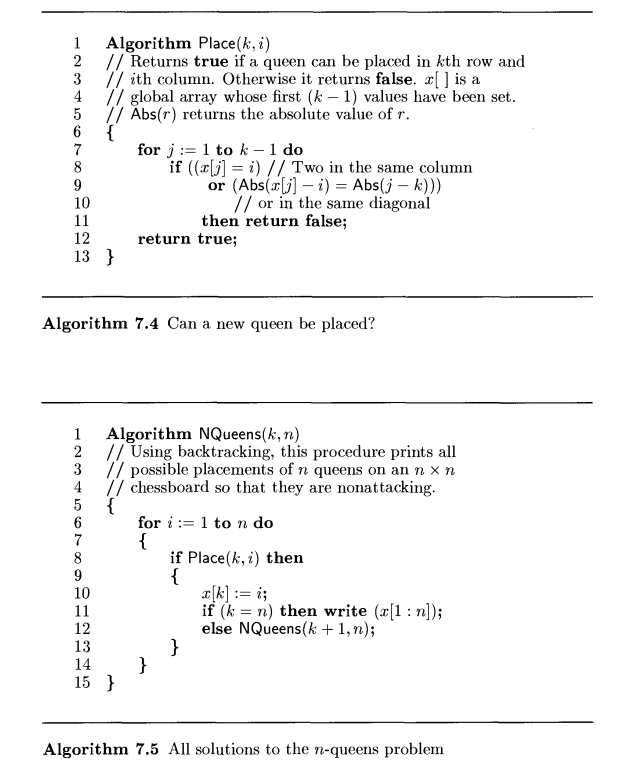
**(n = 4)**



**State Space tree for 4-Queens**



**Algorithm:**



**Time Complexity:** O(n!) (where n = number of queens)

**Space Complexity:** O(n2)

**Code:**

#include <stdio.h>

#define sint(x) scanf("%d", &x)

#define new printf("\n")

#define N 20

int mat[N];

int place(int row, int col)

{

for (int i = 1; i <= row - 1; i++)

{

if (mat[i] == col)

return 0;

else if (abs(mat[i] - col) == abs(i - row))

return 0;

}

return 1;

}

void show(int n)

{

new, new;

for (int i = 1; i <= n; i++)

{

printf("\t%d", i);

}

for (int i = 1; i <= n; i++)

{

printf("\n\n%d", i);

for (int j = 1; j <= n; j++)

{

if (mat[i] == j)

printf("\tQ"); // indicates that a queen has been placed

else

printf("\t\_"); // queen not placed

}

};

new, new;

}

void nqueen(int row, int n)

{

for (int col = 1; col <= n; col++)

{

if (place(row, col))

{

mat[row] = col;

if (row == n)

{

show(n);

return;

}

else

nqueen(row + 1, n);

}

}

}

int main(int argc, char const \*argv[])

{

int n;

printf("Enter the size: ");

sint(n);

nqueen(1, n);

return 0;

}

**Output:**

Enter the size:

4

1 2 3 4

1 \_ Q \_ \_

2 \_ \_ \_ Q

3 Q \_ \_ \_

4 \_ \_ Q \_

1 2 3 4

1 \_ \_ Q \_

2 Q \_ \_ \_

3 \_ \_ \_ Q

4 \_ Q \_ \_

Date:

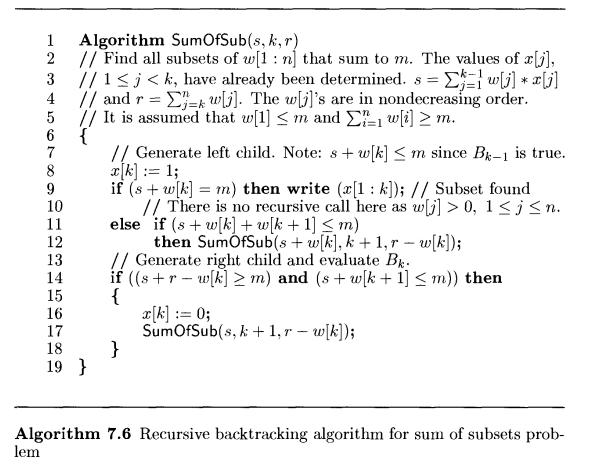
**ii) Sum of Subsets (using Backtracking)**

**Problem Statement:**

Given ‘n’ weights, find the combinations of these weights such that the total weight of the subset equal to ‘m’ (where n and m are given)

W = {5, 7, 10, 12, 15, 18, 20} m = 35

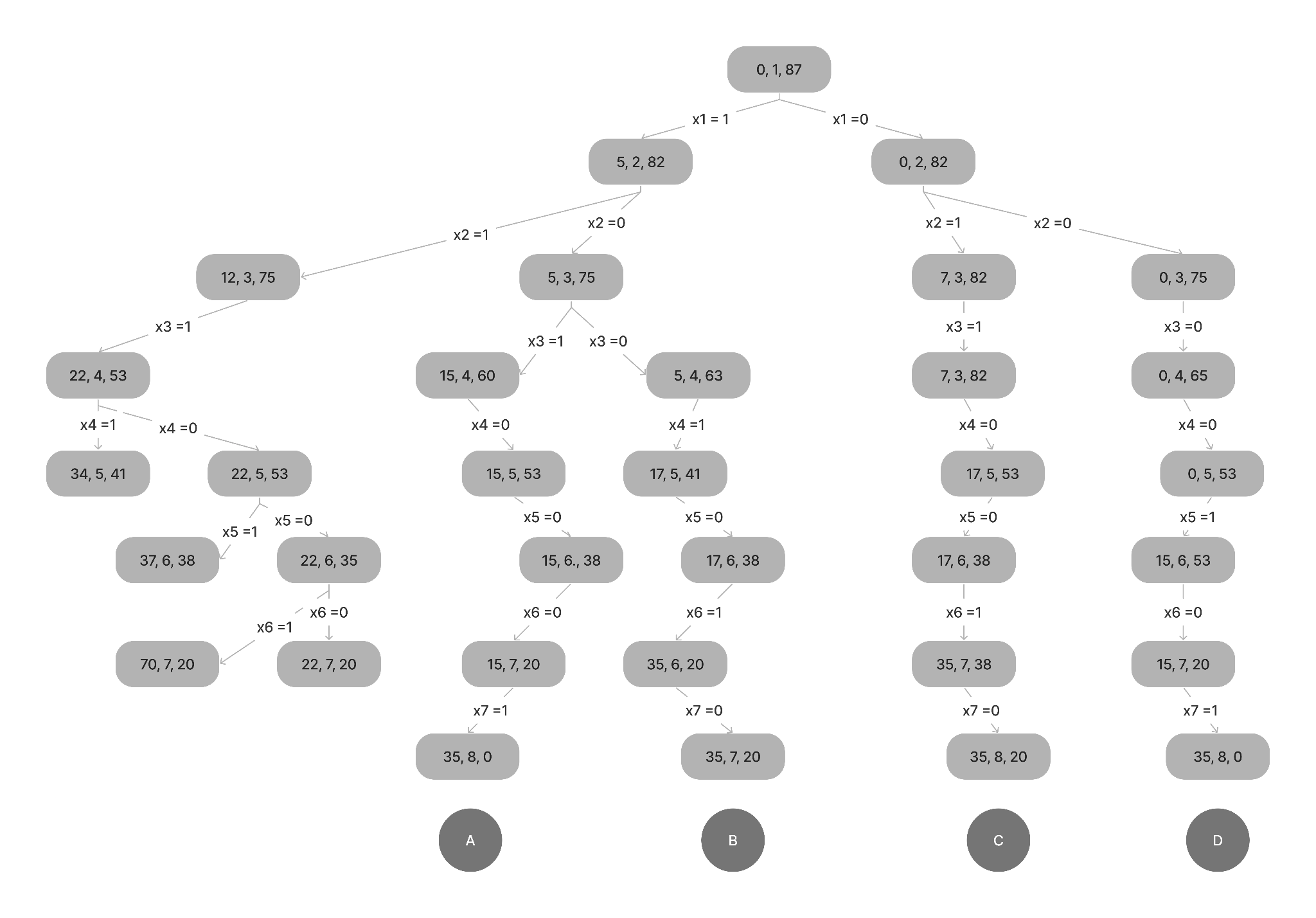
Algorithm:



**Time Complexity:** O(mn)

**Space Complexity:** O(mn)

**State Space tree:**



**Code:**

#include <stdio.h>

#define N 20

#define sint(x) scanf("%d", &x)

int n, m;

void show(int w[N], int k, int x[N])

{

printf("( ");

for (int i = 1; i <= k; i++)

printf("%d, ", x[i]);

printf(") ie. sum of the subset ( ");

for (int i = 1; i <= k; i++)

{

if (x[i] == 1)

printf("%d, ", w[i]);

}

printf(")\n");

}

void sumofsub(int s, int k, int r, int w[], int x[])

{

if (k > n)

return;

x[k] = 1;

// check if adding the first weight itself yields an answer

if (s + w[k] == m)

show(w, k, x);

else if ((s + w[k] + w[k + 1]) <= m)

sumofsub(s + w[k], k + 1, r, w, x);

if (((s + r - w[k]) >= m) && (s + w[k + 1] <= m))

{

x[k] = 0; // exclude the current element

sumofsub(s, k + 1, r, w, x);

}

}

int main(int argc, char const \*argv[])

{

int r = 0; // m= target sum, n = number of elements, r = sum of all elements

printf("Enter the numberof elements: "), sint(n);

int w[n + 1], x[n + 1]; // w is the array of elements (weights), x is the solution array

printf("Enter the elements\n");

for (int i = 1; i <= n; i++)

{

x[i] = 0;

sint(w[i]);

r += w[i];

}

printf("Enter max weight: "), sint(m);

sumofsub(0, 1, r, w, x);

return 0;

**Output:**

c:\P Jeevesh Naidu\college\second year\lV sem\madf codes>cd "c:\P Jeevesh Naidu\college\second year\lV sem\madf codes\" && gcc sumofsubsets.c -o sumofsubsets && "c:\P Jeevesh Naidu\college\second year\lV sem\madf codes\"sumofsubsets

Enter the numberof elements: 7

Enter the elements

5

7

10

12

15

18

20

Enter max weight: 35

( 1, 0, 1, 0, 0, 0, 1, ) ie. sum of the subset ( 5, 10, 20, )

( 1, 0, 0, 1, 0, 1, ) ie. sum of the subset ( 5, 12, 18, )

( 0, 1, 1, 0, 0, 1, ) ie. sum of the subset ( 7, 10, 18, )

( 0, 0, 0, 0, 1, 0, 1, ) ie. sum of the subset ( 15, 20)

Date:

**iii) M-Coloring**

**Problem Statement:**

Write a c program to implement M-COLOURING ALGORITHM on following problem:

Given below are the time slots of taxis. Only one taxi is available for the service at a time. Find the most efficient way to schedule the appointment:

1

2

3

4

5

6

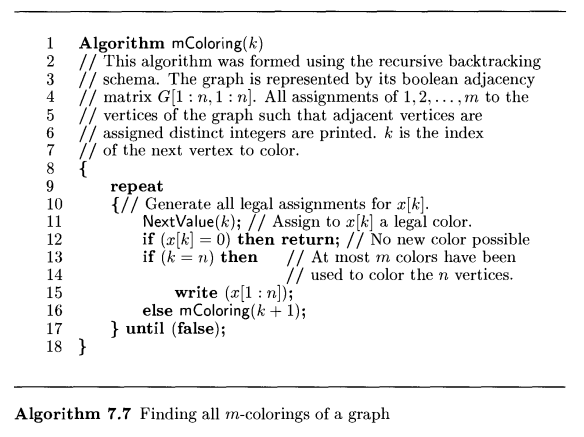
7

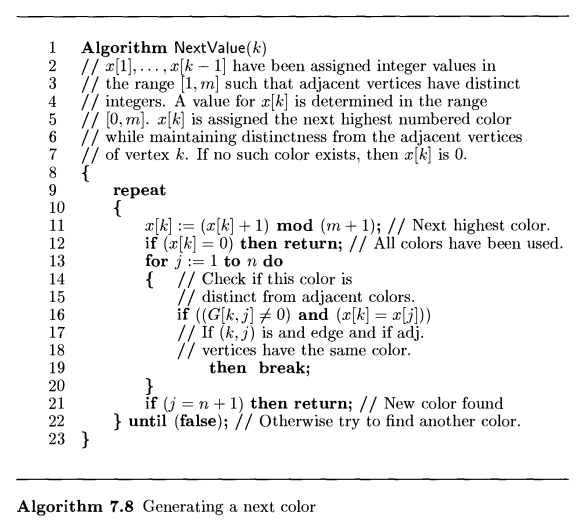
8

9

10

**ALGORITHM :**





**Time Complexity:** O(nmn)

**Space Complexity:** O(mn)

**Code:**

#include <stdio.h>

#define N 20

int G[N][N], count = 0,i,j;

int c[7] = {'V', 'I', 'B', 'G', 'Y', 'O', 'R'};

void nextval(int x[N], int nodes, int colors, int k)

{

do

{

x[k] = (x[k] + 1) % (colors + 1);

if (!x[k])

return;

int j;

for (j = 1; j <= nodes; j++)

{

if ((G[k][j] != 0) && (x[k] == x[j]))

break;

}

if (j == nodes + 1)

return;

} while (1);

}

void show(int x[N], int nodes)

{

printf("%d.\t", ++count);

for (i = 1; i <= nodes; i++)

printf("%c ", c[x[i]]);

printf("\n");

}

void color(int x[N], int nodes, int colors, int k)

{

do

{

nextval(x, nodes, colors, k);

if (!x[k])

return;

if (k == nodes)

show(x, nodes);

else

color(x, nodes, colors, k + 1);

} while (1);

}

int main()

{

int nodes, edges, colors, o, d, x[N];

printf("Enter the number of nodes: "), scanf("%d",&nodes);

for ( i = 1; i <= nodes; i++)

for ( j = 1; j <= nodes; j++)

G[i][j] = 0;

printf("Enter the number of edges: "),scanf("%d",&edges) ;

printf("Enter origin and destination:\n");

for (i = 0; i < edges; i++)

scanf("%d",&o) ,scanf("%d",&d), G[o][d] = G[d][o] = 1;

printf("Enter the number of colors: "), scanf("%d",&colors);

for ( i = 1; i <= nodes; i++)

x[i] = 0;

printf("\n\n");

printf("The following are the possible ways to color the nodes of the graph:");

printf("\n\n");

color(x, nodes, colors, 1);

return 0;

}

**Output:**

c:\P Jeevesh Naidu\college\second year\lV sem\madf codes>cd "c:\P Jeevesh Naidu\college\second year\lV sem\madf codes\" && gcc mcolouring.c -o mcolouring && "c:\P Jeevesh Naidu\college\second year\lV sem\madf codes\"mcolouring

Enter the number of nodes: 10

Enter the number of edges: 10

Enter origin and destination

1 2

1 3

1 4

2 3

3 4

5 6

5 7

6 7

8 9

9 10

Enter the number of colors: 3

The possible ways to color the nodes of the graph are

1.      I B G B I B G I B I

2.      I B G B I B G I B G

3.      I B G B I B G I G I

4.      I B G B I B G I G B

5.      I B G B I B G B I B

6.      I B G B I B G B I G

7.      I B G B I B G B G I

8.      I B G B I B G B G B

9.      I B G B I B G G I B

10.     I B G B I B G G I G

11.     I B G B I B G G B I

12.     I B G B I B G G B G

13.     I B G B I G B I B I

14.     I B G B I G B I B G

15.     I B G B I G B I G I

16.     I B G B I G B I G B

17.     I B G B I G B B I B

18.     I B G B I G B B I G

19.     I B G B I G B B G I

20.     I B G B I G B B G B

21.     I B G B I G B G I B

22.     I B G B I G B G I G

23.     I B G B I G B G B I

24.     I B G B I G B G B G

25.     I B G B B I G I B I

26.     I B G B B I G I B G

27.     I B G B B I G I G I

28.     I B G B B I G I G B

29.     I B G B B I G B I B

30.     I B G B B I G B I G

31.     I B G B B I G B G I

32.     I B G B B I G B G B

33.     I B G B B I G G I B

34.     I B G B B I G G I G

35.     I B G B B I G G B I

36.     I B G B B I G G B G

37.     I B G B B G I I B I

38.     I B G B B G I I B G

39.     I B G B B G I I G I

40.     I B G B B G I I G B

41.     I B G B B G I B I B

42.     I B G B B G I B I G

43.     I B G B B G I B G I

44.     I B G B B G I B G B

45.     I B G B B G I G I B

46.     I B G B B G I G I G

47.     I B G B B G I G B I

48.     I B G B B G I G B G

49.     I B G B G I B I B I

50.     I B G B G I B I B G

51.     I B G B G I B I G I

52.     I B G B G I B I G B

53.     I B G B G I B B I B

54.     I B G B G I B B I G

55.     I B G B G I B B G I

56.     I B G B G I B B G B

57.     I B G B G I B G I B

58.     I B G B G I B G I G

59.     I B G B G I B G B I

60.     I B G B G I B G B G

61.     I B G B G B I I B I

62.     I B G B G B I I B G

63.     I B G B G B I I G I

64.     I B G B G B I I G B

65.     I B G B G B I B I B

66.     I B G B G B I B I G

67.     I B G B G B I B G I

68.     I B G B G B I B G B

69.     I B G B G B I G I B

70.     I B G B G B I G I G

71.     I B G B G B I G B I

72.     I B G B G B I G B G

73.     I G B G I B G I B I

74.     I G B G I B G I B G

75.     I G B G I B G I G I

76.     I G B G I B G I G B

77.     I G B G I B G B I B

78.     I G B G I B G B I G

79.     I G B G I B G B G I

80.     I G B G I B G B G B

81.     I G B G I B G G I B

82.     I G B G I B G G I G

83.     I G B G I B G G B I

84.     I G B G I B G G B G

85.     I G B G I G B I B I

86.     I G B G I G B I B G

87.     I G B G I G B I G I

88.     I G B G I G B I G B

89.     I G B G I G B B I B

90.     I G B G I G B B I G

91.     I G B G I G B B G I

92.     I G B G I G B B G B

93.     I G B G I G B G I B

94.     I G B G I G B G I G

95.     I G B G I G B G B I

96.     I G B G I G B G B G

97.     I G B G B I G I B I

98.     I G B G B I G I B G

99.     I G B G B I G I G I

100.    I G B G B I G I G B

101.    I G B G B I G B I B

102.    I G B G B I G B I G

103.    I G B G B I G B G I

104.    I G B G B I G B G B

105.    I G B G B I G G I B

106.    I G B G B I G G I G

107.    I G B G B I G G B I

108.    I G B G B I G G B G

109.    I G B G B G I I B I

110.    I G B G B G I I B G

111.    I G B G B G I I G I

112.    I G B G B G I I G B

113.    I G B G B G I B I B

114.    I G B G B G I B I G

115.    I G B G B G I B G I

116.    I G B G B G I B G B

117.    I G B G B G I G I B

118.    I G B G B G I G I G

119.    I G B G B G I G B I

120.    I G B G B G I G B G

121.    I G B G G I B I B I

122.    I G B G G I B I B G

123.    I G B G G I B I G I

124.    I G B G G I B I G B

125.    I G B G G I B B I B

126.    I G B G G I B B I G

127.    I G B G G I B B G I

128.    I G B G G I B B G B

129.    I G B G G I B G I B

130.    I G B G G I B G I G

131.    I G B G G I B G B I

132.    I G B G G I B G B G

133.    I G B G G B I I B I

134.    I G B G G B I I B G

135.    I G B G G B I I G I

136.    I G B G G B I I G B

137.    I G B G G B I B I B

138.    I G B G G B I B I G

139.    I G B G G B I B G I

140.    I G B G G B I B G B

141.    I G B G G B I G I B

142.    I G B G G B I G I G

143.    I G B G G B I G B I

144.    I G B G G B I G B G

145.    B I G I I B G I B I

146.    B I G I I B G I B G

147.    B I G I I B G I G I

148.    B I G I I B G I G B

149.    B I G I I B G B I B

150.    B I G I I B G B I G

151.    B I G I I B G B G I

152.    B I G I I B G B G B

153.    B I G I I B G G I B

154.    B I G I I B G G I G

155.    B I G I I B G G B I

156.    B I G I I B G G B G

157.    B I G I I G B I B I

158.    B I G I I G B I B G

159.    B I G I I G B I G I

160.    B I G I I G B I G B

161.    B I G I I G B B I B

162.    B I G I I G B B I G

163.    B I G I I G B B G I

164.    B I G I I G B B G B

165.    B I G I I G B G I B

166.    B I G I I G B G I G

167.    B I G I I G B G B I

168.    B I G I I G B G B G

169.    B I G I B I G I B I

170.    B I G I B I G I B G

171.    B I G I B I G I G I

172.    B I G I B I G I G B

173.    B I G I B I G B I B

174.    B I G I B I G B I G

175.    B I G I B I G B G I

176.    B I G I B I G B G B

177.    B I G I B I G G I B

178.    B I G I B I G G I G

179.    B I G I B I G G B I

180.    B I G I B I G G B G

181.    B I G I B G I I B I

182.    B I G I B G I I B G

183.    B I G I B G I I G I

184.    B I G I B G I I G B

185.    B I G I B G I B I B

186.    B I G I B G I B I G

187.    B I G I B G I B G I

188.    B I G I B G I B G B

189.    B I G I B G I G I B

190.    B I G I B G I G I G

191.    B I G I B G I G B I

192.    B I G I B G I G B G

193.    B I G I G I B I B I

194.    B I G I G I B I B G

195.    B I G I G I B I G I

196.    B I G I G I B I G B

197.    B I G I G I B B I B

198.    B I G I G I B B I G

199.    B I G I G I B B G I

200.    B I G I G I B B G B

201.    B I G I G I B G I B

202.    B I G I G I B G I G

203.    B I G I G I B G B I

204.    B I G I G I B G B G

205.    B I G I G B I I B I

206.    B I G I G B I I B G

207.    B I G I G B I I G I

208.    B I G I G B I I G B

209.    B I G I G B I B I B

210.    B I G I G B I B I G

211.    B I G I G B I B G I

212.    B I G I G B I B G B

213.    B I G I G B I G I B

214.    B I G I G B I G I G

215.    B I G I G B I G B I

216.    B I G I G B I G B G

217.    B G I G I B G I B I

218.    B G I G I B G I B G

219.    B G I G I B G I G I

220.    B G I G I B G I G B

221.    B G I G I B G B I B

222.    B G I G I B G B I G

223.    B G I G I B G B G I

224.    B G I G I B G B G B

225.    B G I G I B G G I B

226.    B G I G I B G G I G

227.    B G I G I B G G B I

228.    B G I G I B G G B G

229.    B G I G I G B I B I

230.    B G I G I G B I B G

231.    B G I G I G B I G I

232.    B G I G I G B I G B

233.    B G I G I G B B I B

234.    B G I G I G B B I G

235.    B G I G I G B B G I

236.    B G I G I G B B G B

237.    B G I G I G B G I B

238.    B G I G I G B G I G

239.    B G I G I G B G B I

240.    B G I G I G B G B G

241.    B G I G B I G I B I

242.    B G I G B I G I B G

243.    B G I G B I G I G I

244.    B G I G B I G I G B

245.    B G I G B I G B I B

246.    B G I G B I G B I G

247.    B G I G B I G B G I

248.    B G I G B I G B G B

249.    B G I G B I G G I B

250.    B G I G B I G G I G

251.    B G I G B I G G B I

252.    B G I G B I G G B G

253.    B G I G B G I I B I

254.    B G I G B G I I B G

255.    B G I G B G I I G I

256.    B G I G B G I I G B

257.    B G I G B G I B I B

258.    B G I G B G I B I G

259.    B G I G B G I B G I

260.    B G I G B G I B G B

261.    B G I G B G I G I B

262.    B G I G B G I G I G

263.    B G I G B G I G B I

264.    B G I G B G I G B G

265.    B G I G G I B I B I

266.    B G I G G I B I B G

267.    B G I G G I B I G I

268.    B G I G G I B I G B

269.    B G I G G I B B I B

270.    B G I G G I B B I G

271.    B G I G G I B B G I

272.    B G I G G I B B G B

273.    B G I G G I B G I B

274.    B G I G G I B G I G

275.    B G I G G I B G B I

276.    B G I G G I B G B G

277.    B G I G G B I I B I

278.    B G I G G B I I B G

279.    B G I G G B I I G I

280.    B G I G G B I I G B

281.    B G I G G B I B I B

282.    B G I G G B I B I G

283.    B G I G G B I B G I

284.    B G I G G B I B G B

285.    B G I G G B I G I B

286.    B G I G G B I G I G

287.    B G I G G B I G B I

288.    B G I G G B I G B G

289.    G I B I I B G I B I

290.    G I B I I B G I B G

291.    G I B I I B G I G I

292.    G I B I I B G I G B

293.    G I B I I B G B I B

294.    G I B I I B G B I G

295.    G I B I I B G B G I

296.    G I B I I B G B G B

297.    G I B I I B G G I B

298.    G I B I I B G G I G

299.    G I B I I B G G B I

300.    G I B I I B G G B G

301.    G I B I I G B I B I

302.    G I B I I G B I B G

303.    G I B I I G B I G I

304.    G I B I I G B I G B

305.    G I B I I G B B I B

306.    G I B I I G B B I G

307.    G I B I I G B B G I

308.    G I B I I G B B G B

309.    G I B I I G B G I B

310.    G I B I I G B G I G

311.    G I B I I G B G B I

312.    G I B I I G B G B G

313.    G I B I B I G I B I

314.    G I B I B I G I B G

315.    G I B I B I G I G I

316.    G I B I B I G I G B

317.    G I B I B I G B I B

318.    G I B I B I G B I G

319.    G I B I B I G B G I

320.    G I B I B I G B G B

321.    G I B I B I G G I B

322.    G I B I B I G G I G

323.    G I B I B I G G B I

324.    G I B I B I G G B G

325.    G I B I B G I I B I

326.    G I B I B G I I B G

327.    G I B I B G I I G I

328.    G I B I B G I I G B

329.    G I B I B G I B I B

330.    G I B I B G I B I G

331.    G I B I B G I B G I

332.    G I B I B G I B G B

333.    G I B I B G I G I B

334.    G I B I B G I G I G

335.    G I B I B G I G B I

336.    G I B I B G I G B G

337.    G I B I G I B I B I

338.    G I B I G I B I B G

339.    G I B I G I B I G I

340.    G I B I G I B I G B

341.    G I B I G I B B I B

342.    G I B I G I B B I G

343.    G I B I G I B B G I

344.    G I B I G I B B G B

345.    G I B I G I B G I B

346.    G I B I G I B G I G

347.    G I B I G I B G B I

348.    G I B I G I B G B G

349.    G I B I G B I I B I

350.    G I B I G B I I B G

351.    G I B I G B I I G I

352.    G I B I G B I I G B

353.    G I B I G B I B I B

354.    G I B I G B I B I G

355.    G I B I G B I B G I

356.    G I B I G B I B G B

357.    G I B I G B I G I B

358.    G I B I G B I G I G

359.    G I B I G B I G B I

360.    G I B I G B I G B G

361.    G B I B I B G I B I

362.    G B I B I B G I B G

363.    G B I B I B G I G I

364.    G B I B I B G I G B

365.    G B I B I B G B I B

366.    G B I B I B G B I G

367.    G B I B I B G B G I

368.    G B I B I B G B G B

369.    G B I B I B G G I B

370.    G B I B I B G G I G

371.    G B I B I B G G B I

372.    G B I B I B G G B G

373.    G B I B I G B I B I

374.    G B I B I G B I B G

375.    G B I B I G B I G I

376.    G B I B I G B I G B

377.    G B I B I G B B I B

378.    G B I B I G B B I G

379.    G B I B I G B B G I

380.    G B I B I G B B G B

381.    G B I B I G B G I B

382.    G B I B I G B G I G

383.    G B I B I G B G B I

384.    G B I B I G B G B G

385.    G B I B B I G I B I

386.    G B I B B I G I B G

387.    G B I B B I G I G I

388.    G B I B B I G I G B

389.    G B I B B I G B I B

390.    G B I B B I G B I G

391.    G B I B B I G B G I

392.    G B I B B I G B G B

393.    G B I B B I G G I B

394.    G B I B B I G G I G

395.    G B I B B I G G B I

396.    G B I B B I G G B G

397.    G B I B B G I I B I

398.    G B I B B G I I B G

399.    G B I B B G I I G I

400.    G B I B B G I I G B

401.    G B I B B G I B I B

402.    G B I B B G I B I G

403.    G B I B B G I B G I

404.    G B I B B G I B G B

405.    G B I B B G I G I B

406.    G B I B B G I G I G

407.    G B I B B G I G B I

408.    G B I B B G I G B G

409.    G B I B G I B I B I

410.    G B I B G I B I B G

411.    G B I B G I B I G I

412.    G B I B G I B I G B

413.    G B I B G I B B I B

414.    G B I B G I B B I G

415.    G B I B G I B B G I

416.    G B I B G I B B G B

417.    G B I B G I B G I B

418.    G B I B G I B G I G

419.    G B I B G I B G B I

420.    G B I B G I B G B G

421.    G B I B G B I I B I

422.    G B I B G B I I B G

423.    G B I B G B I I G I

424.    G B I B G B I I G B

425.    G B I B G B I B I B

426.    G B I B G B I B I G

427.    G B I B G B I B G I

428.    G B I B G B I B G B

429.    G B I B G B I G I B

430.    G B I B G B I G I G

431.    G B I B G B I G B I

432.    G B I B G B I G B G

--------------------------------

Process exited after 6.661 seconds with return value 0

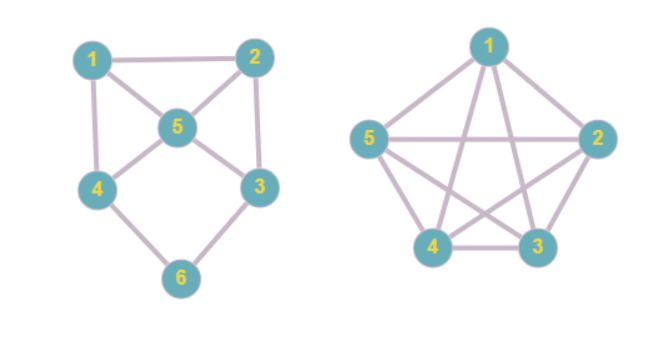
Press any key to continue .

Date:

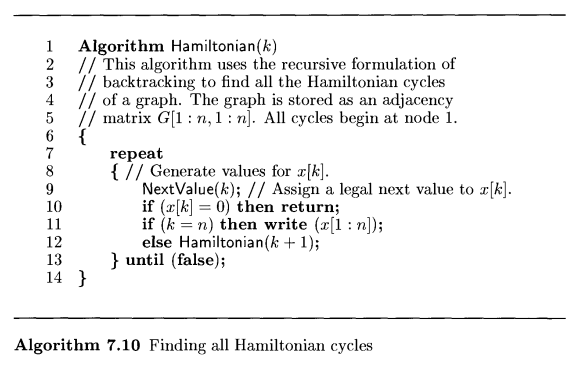
Hamiltonian Cycles

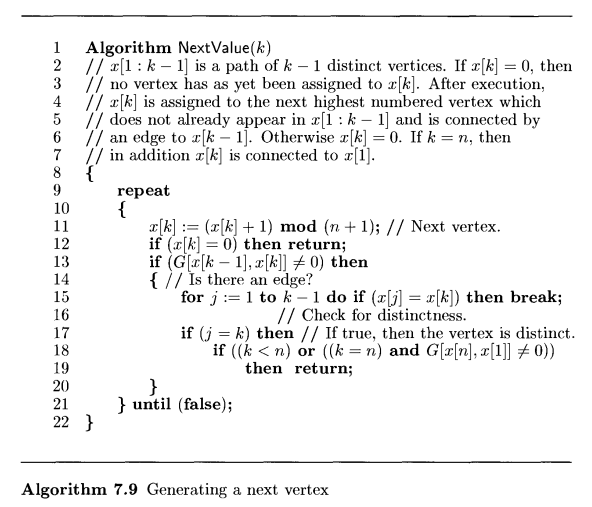
**Problem Statement:**

Find the Hamiltonian cycles in the given graph using the backtracking approach



**Algorithm:**





**Time Complexity:** O(n!)

**Space Complexity**: O(n)

**Code:**

#include <stdio.h>

#define sint(x) scanf("%d", &x)

#define N 100

#define line printf("\n--------------------------------------------------------\n")

int G[N][N], count = 0;

void show(int x[N], int nodes)

{

printf("%d.\t", ++count);

for (int i = 1; i <= nodes; i++)

printf("%d -> ", x[i]);

printf("%d", x[1]);

printf("\n");

}

void nextval(int x[N], int nodes, int k)

{

do

{

// printf("nextval do while\n");

x[k] = (x[k] + 1) % (nodes + 1); // next vertex

// printf("xk val %d\n", x[k]);

if (!x[k])

return;

// printf("x[k]!=0\n");

// printf("g val %d\n", G[x[k - 1]][x[k]]);

if (G[x[k - 1]][x[k]] != 0)

{

int j;

for (j = 1; j <= k - 1; j++)

if (x[j] == x[k])

break;

if (j == k)

if ((k < nodes) || ((k == nodes && G[x[nodes]][x[1]] != 0)))

return;

}

} while (1);

}

void ham(int x[N], int nodes, int k)

{

do

{

// printf("ham do while\n");

// Generate values for x[k]

nextval(x, nodes, k);

if (!x[k])

return;

if (k == nodes)

show(x, nodes);

else

ham(x, nodes, k + 1);

} while (1);

}

int main()

{

int nodes, edges, o, d, x[N];

printf("Enter the number of nodes: "), sint(nodes);

for (int i = 1; i <= nodes; i++)

for (int j = 1; j <= nodes; j++)

G[i][j] = 0;

printf("Enter the number of edges: "), sint(edges);

printf("Enter origin and destination\n");

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

sint(o), sint(d), G[o][d] = G[d][o] = 1;

int k;

printf("Enter the starting vertex: "), sint(k);

x[1] = k; // here we can assign '1' if we always want to start from vertex 1 or we can assign k value to start from a different vertex

for (int i = 2; i <= nodes; i++)

x[i] = 0;

line, printf("The possible Hamiltonian Cycles starting from node %d are", k), line;

ham(x, nodes, 2);

return 0;

}

Output:

1]

C:\P Jeevesh Naidu\college\second year\lV sem\madf codes>cd "c:\P Jeevesh Naidu\college\second year\lV sem\madf codes\" && gcc hamiltonian.c -o hamiltonian && "c:\P Jeevesh Naidu\college\second year\lV sem\madf codes\"hamiltonian

Enter the number of nodes: 6

Enter the number of edges: 10

Enter origin and destination

1 2

1 3

1 4

2 3

2 5

3 4

3 5

4 5

4 6

5 6

Enter the starting vertex: 1

--------------------------------------------------------

The possible Hamiltonian Cycles starting from node 1 are

--------------------------------------------------------

1. 1 -> 2 -> 3 -> 5 -> 6 -> 4 -> 1

2. 1 -> 2 -> 5 -> 6 -> 4 -> 3 -> 1

3. 1 -> 3 -> 2 -> 5 -> 6 -> 4 -> 1

4. 1 -> 3 -> 4 -> 6 -> 5 -> 2 -> 1

5. 1 -> 4 -> 6 -> 5 -> 2 -> 3 -> 1

6. 1 -> 4 -> 6 -> 5 -> 3 -> 2 -> 1

2]

c:\P Jeevesh Naidu\college\second year\lV sem\madf codes>cd "c:\P Jeevesh Naidu\college\second year\lV sem\madf codes\" && gcc hamiltonian.c -o hamiltonian && "c:\P Jeevesh Naidu\college\second year\lV sem\madf codes\"hamiltonian

Enter the number of nodes: 5

Enter the number of edges: 10

Enter origin and destination

1 2

1 3

1 4

1 5

2 3

2 4

2 5

3 4

3 5

4 5

Enter the starting vertex: 1

--------------------------------------------------------

The possible Hamiltonian Cycles starting from node 1 are

--------------------------------------------------------

1. 1 -> 2 -> 3 -> 4 -> 5 -> 1

2. 1 -> 2 -> 3 -> 5 -> 4 -> 1

3. 1 -> 2 -> 4 -> 3 -> 5 -> 1

4. 1 -> 2 -> 4 -> 5 -> 3 -> 1

5. 1 -> 2 -> 5 -> 3 -> 4 -> 1

6. 1 -> 2 -> 5 -> 4 -> 3 -> 1

7. 1 -> 3 -> 2 -> 4 -> 5 -> 1

8. 1 -> 3 -> 2 -> 5 -> 4 -> 1

9. 1 -> 3 -> 4 -> 2 -> 5 -> 1

10. 1 -> 3 -> 4 -> 5 -> 2 -> 1

11. 1 -> 3 -> 5 -> 2 -> 4 -> 1

12. 1 -> 3 -> 5 -> 4 -> 2 -> 1

13. 1 -> 4 -> 2 -> 3 -> 5 -> 1

14. 1 -> 4 -> 2 -> 5 -> 3 -> 1

15. 1 -> 4 -> 3 -> 2 -> 5 -> 1

16. 1 -> 4 -> 3 -> 5 -> 2 -> 1

17. 1 -> 4 -> 5 -> 2 -> 3 -> 1

18. 1 -> 4 -> 5 -> 3 -> 2 -> 1

19. 1 -> 5 -> 2 -> 3 -> 4 -> 1

20. 1 -> 5 -> 2 -> 4 -> 3 -> 1

21. 1 -> 5 -> 3 -> 2 -> 4 -> 1

22. 1 -> 5 -> 3 -> 4 -> 2 -> 1

23. 1 -> 5 -> 4 -> 2 -> 3 -> 1

24. 1 -> 5 -> 4 -> 3 -> 2 -> 1

Date:

1. **0/1 KNAPSACK PROBLEM**

**Problem Statement:**

Write a c program to implement 0/1 knapsack using dynamic programming on the following knapsack :

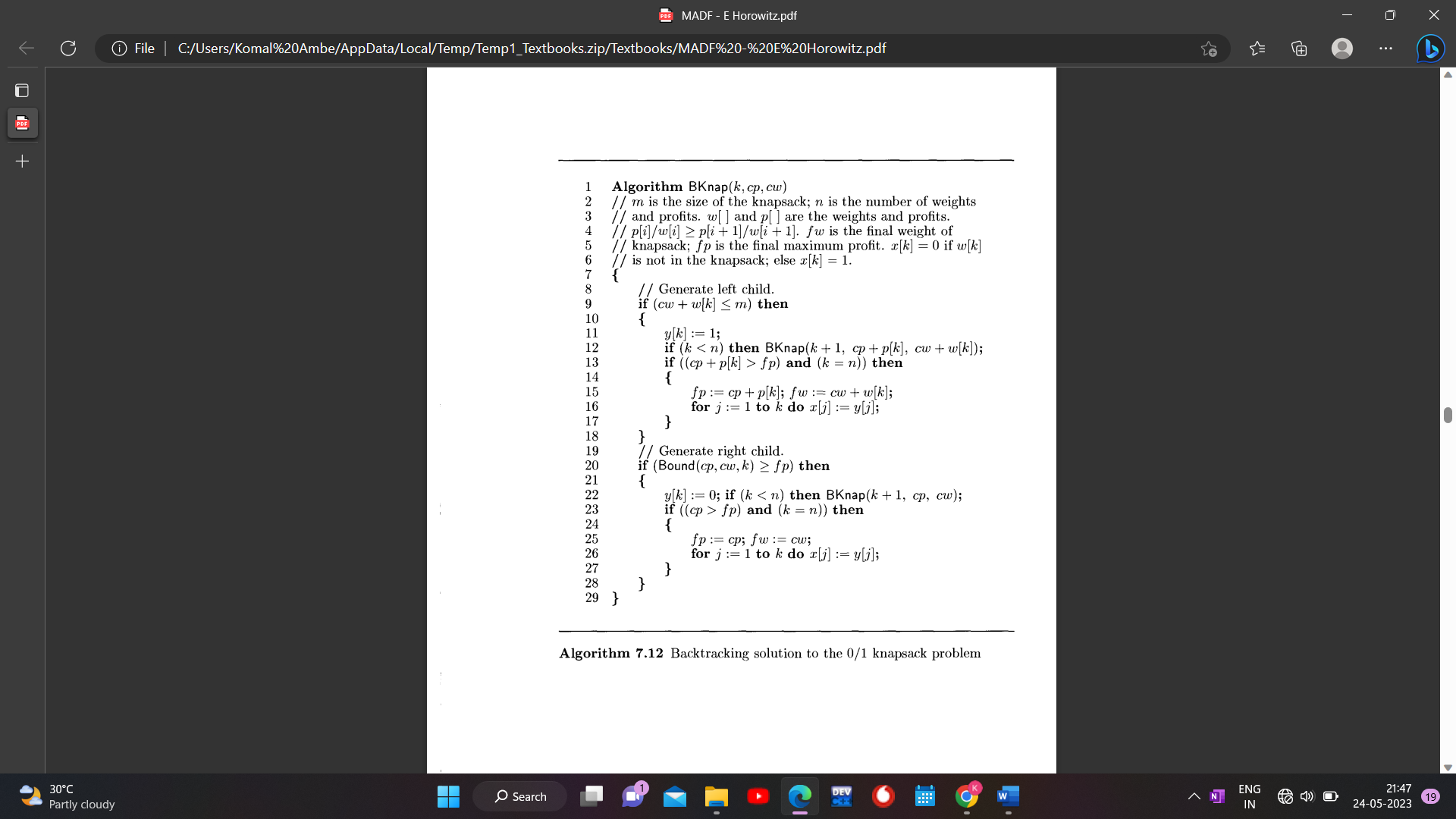
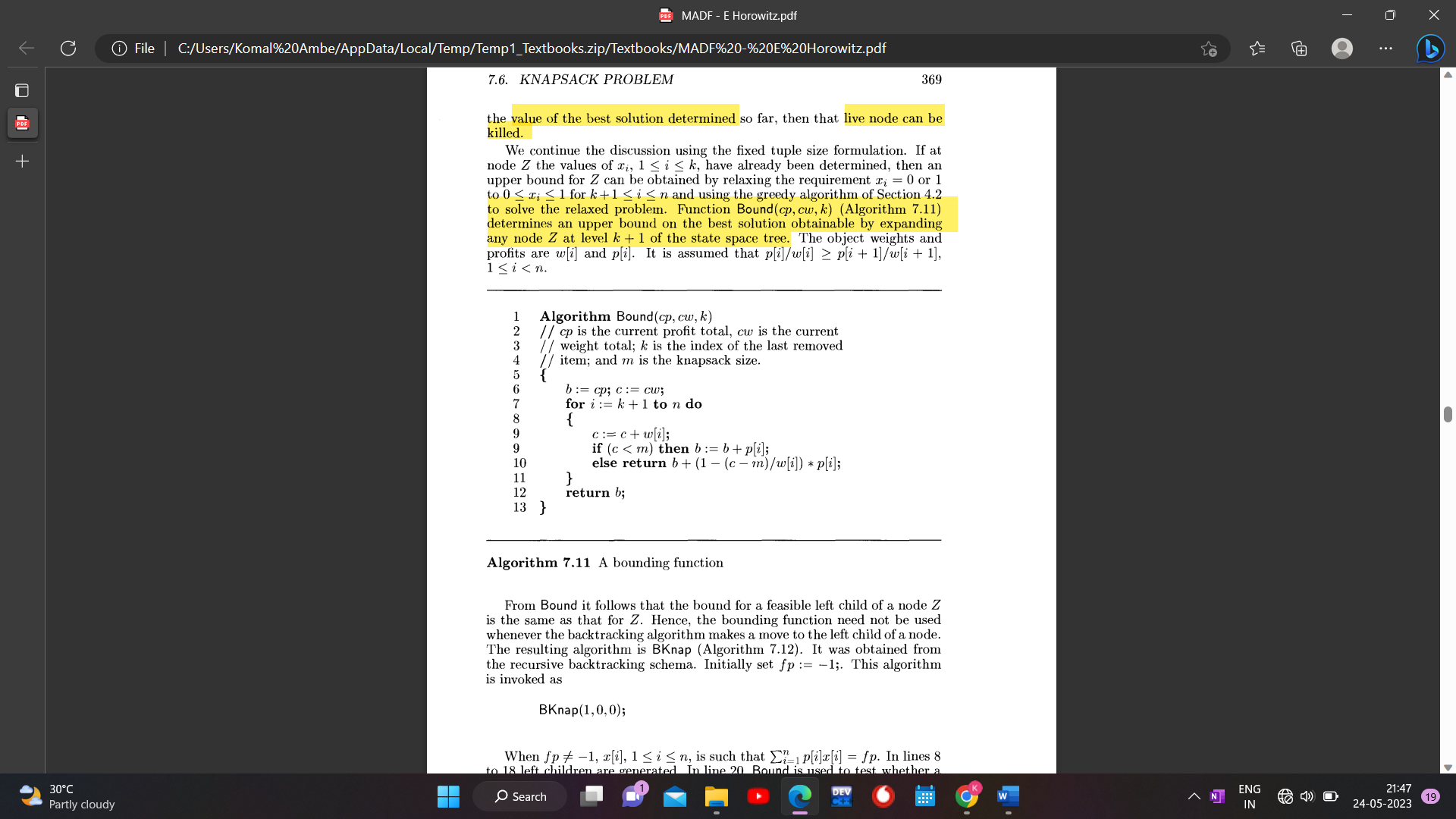
N=4

M=8

p1,p2,p3,p4={3,5,6,10}

w1,w2,w3,w4}={2,3,4,5,}

**Algorithm;**

****

**Time Complexity:** O(2^n)

**Space Complexity**: O(n)

**Code**

#include<stdio.h>

#include<stdlib.h>

int p[100], w[100];

int x[100],y[100];

int m, n,fp=0,fw=0;

void knapsack(int,int,int);

float bound(int,int,int);

void knapsack(int k,int cp,int cw)

{

if(cw+w[k]<=m)

{

y[k]=1;

if(k<n)knapsack(k+1,cp+p[k],cw+w[k]);

if((cp+p[k]>fp) && k==n)

{

fp=cp+p[k];

fw=cw+w[k];

for(int j=1;j<=k;j++)x[j]=y[j];

}

}

if(bound(cp,cw,k)>=fp)

{

y[k]=0;

if(k<n)

knapsack(k+1,cp,cw);

if((cp>fp)&& k==n)

{

fp=cp;

fw=cw;

for(int j=1;j<=k;j++)

x[j]=y[j];

}

}

}

float bound(int cp,int cw,int k)

{

int b=cp,c=cw;

for(int i=k+1;i<=n;i++)

{

c=c+w[i];

if(c<m)

b=b+p[i];

else

return b+(1-(c-m)/(float)w[i])\*p[i];

}

return b;

}

int main()

{

int profit=0, weight=0;

printf("Enter number of objects[n]: ");

scanf("%d",&n);

printf("Enter sack size[m]: ");

scanf("%d",&m);

printf("\nEnter profits :\n");

for(int i=1; i<=n; i++)

{

printf("P[%d]: ",i);

scanf("%d",&p[i]);

}

printf("\nEnter weights : \n");

for(int i=1; i<=n; i++)

{

printf("W[%d]: ",i);

scanf("%d",&w[i]);

x[i] = 0;

}

knapsack(0,0,1);

printf("\nSolution Vector: (");

for(int i=1; i<=n; i++)

{

profit+=p[i]\*x[i];

weight+=w[i]\*x[i];

printf("%d",x[i]);

if(i<n)

{

printf(",");

}

}

printf(")\n");

printf("\nMAX Profit = %d \n",profit);

printf("Weight Occupied = %d ",weight);

return 0;

}

**OUTPUT :**

c:\P Jeevesh Naidu\college\second year\lV sem\madf codes>cd "c:\P Jeevesh Naidu\college\second year\lV sem\madf codes\" && gcc 0/1ks.c -o 0/1ks && "c:\P Jeevesh Naidu\college\second year\lV sem\madf codes\"0/1ks

Enter number of objects[n]: 4

Enter sack size[m]: 8

Enter profits :

P[1]: 3

P[2]: 5

P[3]: 6

P[4]: 10

Enter weights :

W[1]: 2

W[2]: 3

W[3]: 4

W[4]: 5

Solution Vector: (1,0,0,1)

MAX Profit = 13

Weight Occupied = 7

--------------------------------

Process exited after 19.19 seconds with return value 0

Press any key to continue . . .

**Conclusion:**

Several Optimization problems were studied and implemented using the Backtracking Programming Algorithm. N-Queens, Sum of Subset, M-Coloring, Hamiltonian cycles, were implemented using the Backtracking Approach.