DAA Lab-10

22/03/25 – 230962326 – Davasam Karthikeya - AIMLB

Solved question (Binomial coefficient):

```
#include<stdio.h>
#include<stdlib.h>

int c[20][20];

void binomial(int n, int k) {
    for(int i = 0; i <= n; i++)
        for(int j = 0; j <= i && j <= k; j++) {
            if(j == 0 || j == i)
                c[i][j] = 1;
            else
                c[i][j] = c[i - 1][j - 1] + c[i - 1][j];
        }
}

void main() {
    int n,k;

    printf("Enter n: ");
    scanf("%d", &n);
    printf("Enter k: ");
    scanf("%d", &k);

    binomial(n, k);

    printf("Matrix:\n");
    for(int i = 0; i <= n; i++) {
        for(int j = 0; j <= i && j <= k; j++)
            printf("%d ", c[i][j]);
        printf("\n");
    }

    printf("Result: %d\n", c[n][k]);
}
```

Output

```
Enter n: 4
Enter k: 2
Matrix:
1
1 1
1 2 1
1 3 3
1 4 6
Result: 6
```

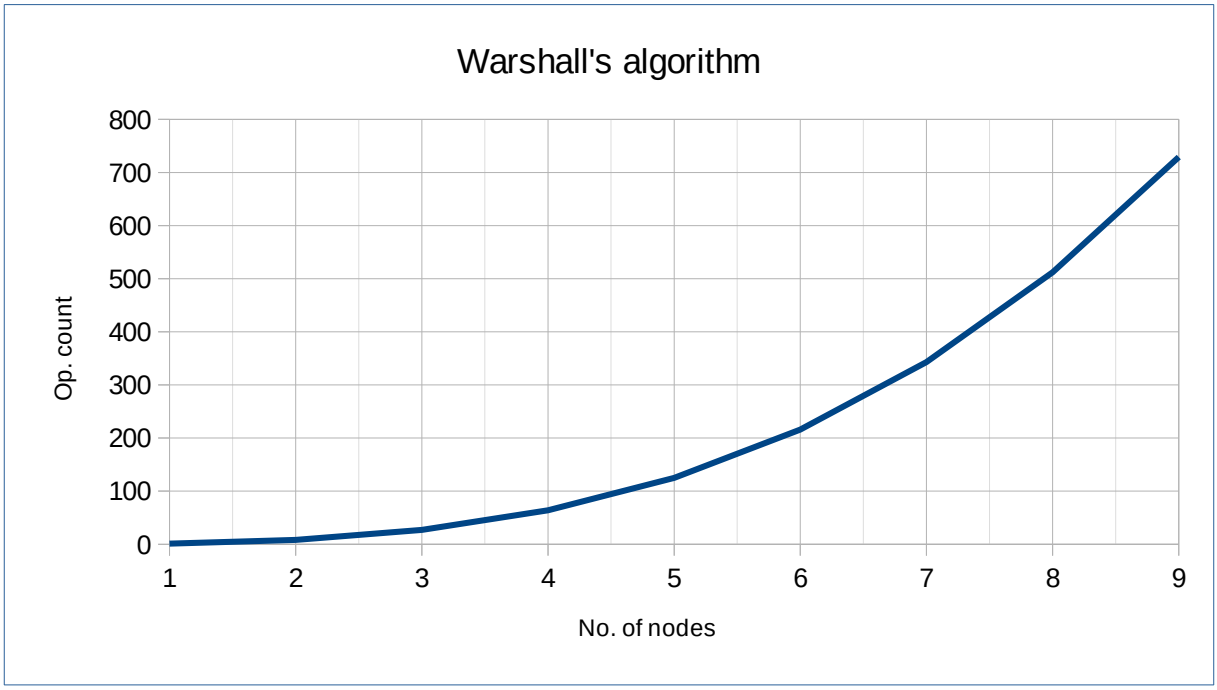
Question 1 (Warshall's algorithm):

```
#include <stdio.h>

#define SIZE 5

void warshall(int graph[][SIZE]) {
    for(int k = 0; k < SIZE; k++)
```

```
        for(int i = 0; i < SIZE; i++)
            for(int j = 0; j < SIZE; j++)
                graph[i][j] = graph[i][j] || (graph[i][k] && graph[k][j]);

    printf("\nTrans. closure:\n");
    for(int i = 0; i < SIZE; i++) {
        for(int j = 0; j < SIZE; j++)
            printf("%d ", graph[i][j]);
        printf("\n");
    }
}

void main() {
    int graph[][SIZE] = {
        {0, 5, 3, 0, 0},
        {0, 0, 2, 6, 0},
        {0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0},
        {8, 0, 0, 0, 0}
    };

    warshall(graph);
}
```

Output

Trans. closure:
0 1 1 1 0
0 0 1 1 0
0 0 0 0 0
0 0 0 0 0
1 1 1 1 0



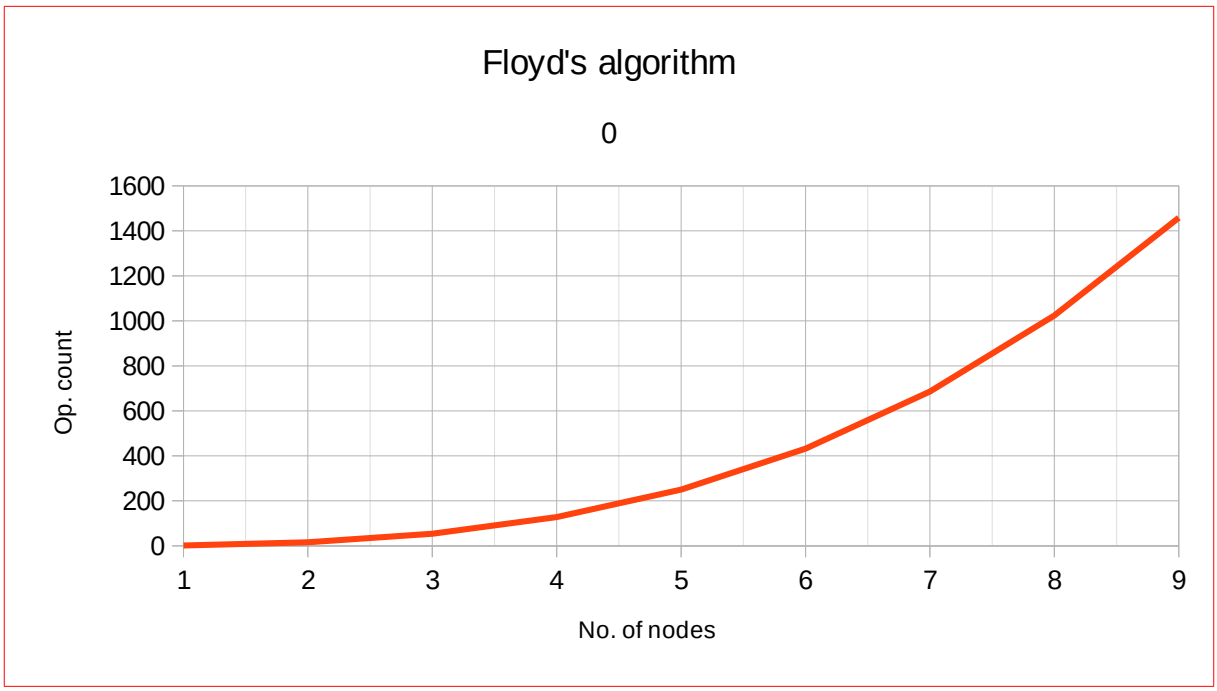
Question 2 (Floyd's algorithm):

```
#include <stdio.h>

#define SIZE 5

void floyd(int graph[][SIZE]) {
    int *arr = (int*) graph;
    for(int i = 0; i < SIZE * SIZE; i++)
```

```
        arr[i] = arr[i] ? arr[i] : 99999999;

    for(int k = 0; k < SIZE; k++)
        for(int i = 0; i < SIZE; i++)
            for(int j = 0; j < SIZE; j++) {
                int temp = graph[i][k] + graph[k][j];
                graph[i][j] = temp < graph[i][j] ? temp : graph[i][j];
            }

    for(int i = 0; i < SIZE * SIZE; i++)
        arr[i] = arr[i] != 99999999 ? arr[i] : 0;

    printf("\nShortest paths:\n");
    for(int i = 0; i < SIZE; i++) {
        for(int j = 0; j < SIZE; j++)
            printf("%d ", graph[i][j]);
        printf("\n");
    }
}

void main() {
    int graph[][SIZE] = {
        {0, 5, 3, 0, 0},
        {0, 0, 2, 6, 0},
        {0, 0, 0, 0, 0},
        {0, 0, 0, 0, 0},
        {8, 0, 0, 0, 0}
    };

    floyd(graph);
}
```

Output

Shortest paths:
0 5 3 11 0
0 0 2 6 0
0 0 0 0 0
0 0 0 0 0
8 13 11 19 0



Question 3 (Bottom-up knapsack):

```
#include <stdio.h>

#define SIZE 3

int knapsack(int W, int val[], int wei[]) {
    int sack[W];
    for(int i = 0; i < W; i++)
        sack[W] = 0;

    for (int i = 0; i < SIZE; i++)
        for (int j = W; j >= wei[i]; j--) {
            int temp = sack[j - wei[i]] + val[i];
            sack[j] = temp > sack[j] ? temp : sack[j];
        }

    return sack[W];
}

void main() {
    int val[SIZE] = {3, 2, 1};
    int wei[SIZE] = {1, 5, 4};
    int W = 5;

    printf("Max value: %d\n", knapsack(W, val, wei));
}
```

Output

Max value: 4