

**Assignment-6**

**Name: - Mann Kothari**

**Reg. No.: - 22BCB7064**

**Course Title: - Design and Analysis of Algorithm (Embedded Lab)**

**Course Code: - CSE3023**

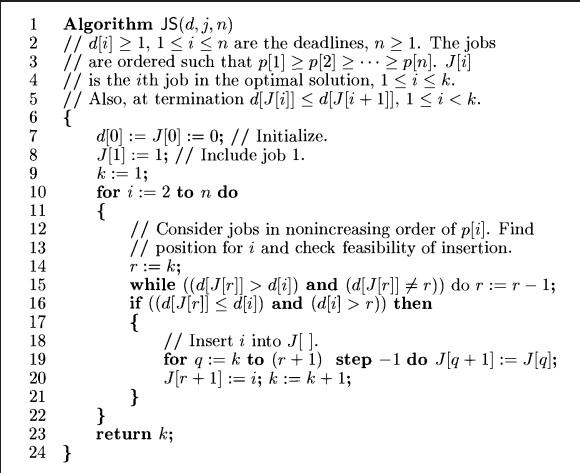
**Slot: - L21+L22**

**Submitted to: - Prof. Tanikella Divya Naga Pavani**

* **INDEX**
* Job Sequencing with deadlines Problem using the Greedy Method.
* Algorithm
* Code
* Output
* Complexity
* Single Source Shortest paths (Dijkstra's algorithm) Problem using the Greedy Method.
* Algorithm
* Code
* Output
* Complexity
* Complexity Chart

1. Write a C/C++ program to Implement the Job Sequencing with deadlines Problem using the Greedy Method.

Algorithm: -



Time Complexity: -

O(n2)

Code: -

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

typedef struct Job

{

    char id;

    int dead;

    int profit;

} Job;

int compare(const void \*a, const void \*b)

{

    Job \*temp1 = (Job \*)a;

    Job \*temp2 = (Job \*)b;

    return (temp2->profit - temp1->profit);

}

int min(int num1, int num2)

{

    return (num1 > num2) ? num2 : num1;

}

void printJobScheduling(Job arr[], int n)

{

    qsort(arr, n, sizeof(Job), compare);

    int result[n];

    bool slot[n];

    for (int i = 0; i < n; i++)

        slot[i] = false;

    for (int i = 0; i < n; i++)

    {

        for (int j = min(n, arr[i].dead) - 1; j >= 0; j--)

        {

            if (slot[j] == false)

            {

                result[j] = i;

                slot[j] = true;

                break;

            }

        }

    }

    for (int i = 0; i < n; i++)

        if (slot[i])

            printf("%c ", arr[result[i]].id);

}

int main()

{

    Job arr[] = {{'a', 2, 100},

                 {'b', 1, 19},

                 {'c', 2, 27},

                 {'d', 1, 25},

                 {'e', 3, 15}};

    int n = sizeof(arr) / sizeof(arr[0]);

    printf(

        "Following is maximum profit sequence of jobs \n");

    printJobScheduling(arr, n);

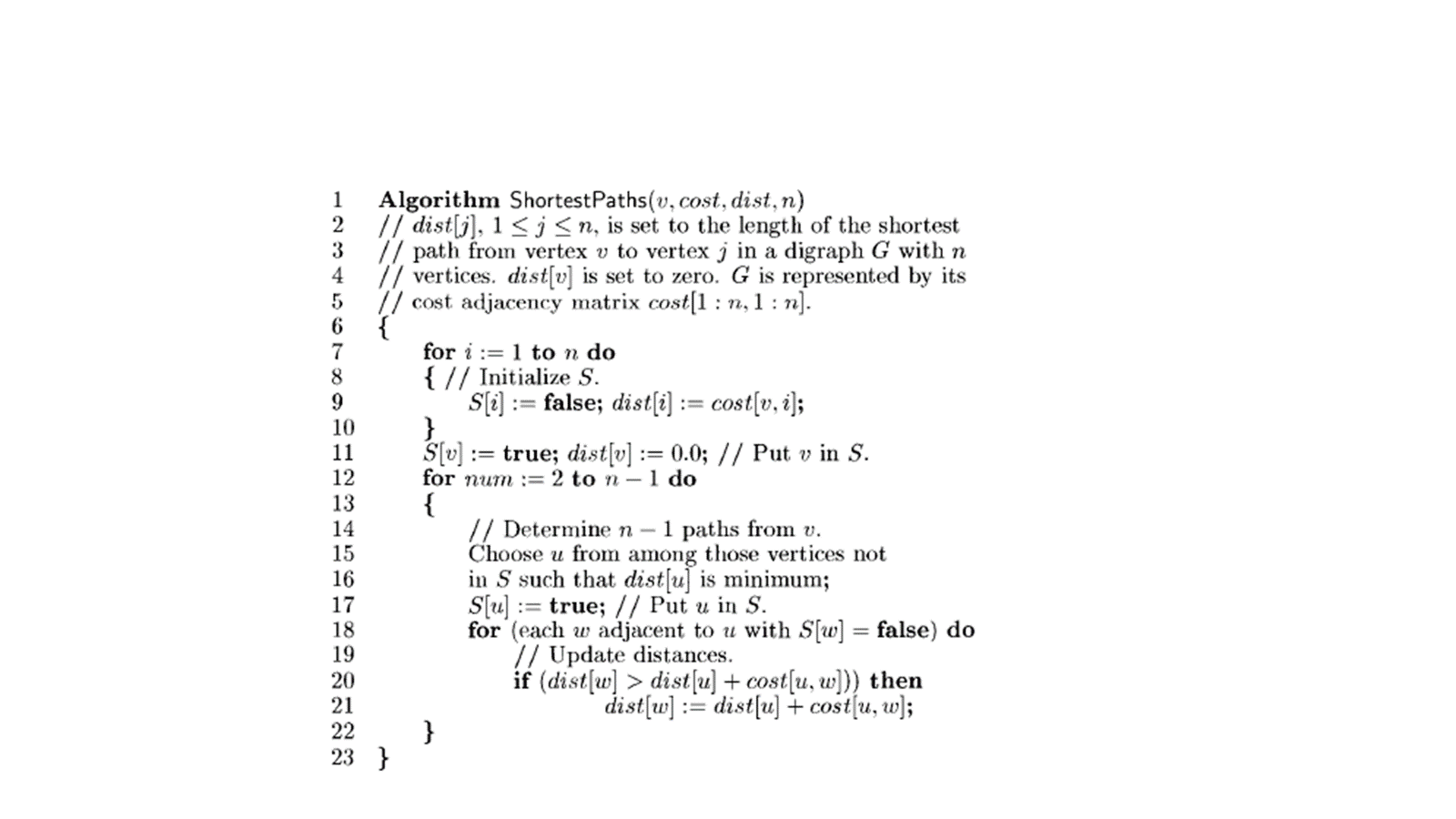
    return 0;

}

Output: -

1. Write a C/C++ program to Implement the Single Source Shortest paths (Dijkstra's algorithm) Problem using the Greedy Method.

Algorithm: -



Time Complexity: -

O(n2)

Code: -

#include <limits.h>

#include <stdbool.h>

#include <stdio.h>

#define V 9

int minDistance(int dist[], bool sptSet[])

{

    int min = INT\_MAX, min\_index;

    for (int v = 0; v < V; v++)

        if (sptSet[v] == false && dist[v] <= min)

            min = dist[v], min\_index = v;

    return min\_index;

}

void printSolution(int dist[])

{

    printf("Vertex \t\t Distance from Source\n");

    for (int i = 0; i < V; i++)

        printf("%d \t\t\t\t %d\n", i, dist[i]);

}

void dijkstra(int graph[V][V], int src)

{

    int dist[V];

    bool sptSet[V];

    for (int i = 0; i < V; i++)

        dist[i] = INT\_MAX, sptSet[i] = false;

    dist[src] = 0;

    for (int count = 0; count < V - 1; count++)

    {

        int u = minDistance(dist, sptSet);

        sptSet[u] = true;

        for (int v = 0; v < V; v++)

            if (!sptSet[v] && graph[u][v] && dist[u] != INT\_MAX && dist[u] + graph[u][v] < dist[v])

                dist[v] = dist[u] + graph[u][v];

    }

    printSolution(dist);

}

int main()

{

    int graph[V][V] = {{0, 4, 0, 0, 0, 0, 0, 8, 0},

                       {4, 0, 8, 0, 0, 0, 0, 11, 0},

                       {0, 8, 0, 7, 0, 4, 0, 0, 2},

                       {0, 0, 7, 0, 9, 14, 0, 0, 0},

                       {0, 0, 0, 9, 0, 10, 0, 0, 0},

                       {0, 0, 4, 14, 10, 0, 2, 0, 0},

                       {0, 0, 0, 0, 0, 2, 0, 1, 6},

                       {8, 11, 0, 0, 0, 0, 1, 0, 7},

                       {0, 0, 2, 0, 0, 0, 6, 7, 0}};

    dijkstra(graph, 0);

    return 0;

}

Output:

|  |  |  |
| --- | --- | --- |
| Algorithm | Time Complexity | Space Complexity |
| Job Sequencing with deadlines Problem using the Greedy Method. | O(n2) | O(n) |
| Single Source Shortest paths (Dijkstra's algorithm) Problem using the Greedy Method. | O(nlogn) | O(n) |