NAME:	Shubham Vishwakarma
UID No.	2021700071
BRANCH:	S.Y CSE-DS
BATCH:	D
SUBJECT	Design and Analysis of Algorithms
EXPERIMENT No.	6
Date of Performance	28/03/2023
Date of Submission	30/03/2023

AIM:	**Prim's Algorithm Implementation**
Program 1	
PROBLEM STATEMENT :	Implement the prim's algorithm for the given graph.
ALGORITHM/ THEORY:	**Step 1:** Determine an arbitrary vertex as the starting vertex of the MST. **Step 2:** Follow steps 3 to 5 till there are vertices that are not included in the MST (known as fringe vertex). **Step 3:** Find edges connecting any tree vertex with the fringe vertices. **Step 4:** Find the minimum among these edges. **Step 5:** Add the chosen edge to the MST if it does not form any cycle. **Step 6:** Return the MST and exit

PROGRAM:

```c
#include <limits.h>
#include <stdbool.h>
#include <stdio.h>

#define V 5

int minKey(int key[], bool mstSet[])
{
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++)
        if (mstSet[v] == false && key[v] < min)
            min = key[v], min_index = v;

    return min_index;
}

int printMST(int parent[], int graph[V][V])
{
    printf("Edge \tWeight\n");
    for (int i = 1; i < V; i++){
        printf("%d - %d \t%d \n", parent[i], i,graph[i][parent[i]]);
    }
}

void primMST(int graph[V][V])
{
    int parent[V];

    int key[V];

    bool mstSet[V];

    for (int i = 0; i < V; i++)
        key[i] = INT_MAX, mstSet[i] = false;

    key[0] = 0;

    parent[0] = -1;

    for (int count = 0; count < V - 1; count++)
    {
        int u = minKey(key, mstSet);

        mstSet[u] = true;

        for (int v = 0; v < V; v++)
```

```
            if (graph[u][v] && mstSet[v] == false
                && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }

    // print the constructed MST
    printMST(parent, graph);
}

int main()
{
    int graph[V][V] = { { 0, 2, 0, 6, 0 },
                        { 2, 0, 3, 8, 5 },
                        { 0, 3, 0, 0, 7 },
                        { 6, 8, 0, 0, 9 },
                        { 0, 5, 7, 9, 0 } };

    // Print the solution
    primMST(graph);

    return 0;
}
```

RESULT:

```
PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp6\output> cd 'c:\Users\smsha\
Desktop\SEM 4\DAA\Practicals\Exp6\output'
PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp6\output> & .\'prims.exe'
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5
PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp6\output>
```

CONCLUSION :	**It does this by using a greedy approach of selecting the minimum cost edge for every vertex**. The time complexity for the algorithm depends on the data structure used to implement it. Prims Algorithm is favored over Krushkal's to find MST when the graph is dense.