D.Y PATIL INTERNATIONAL UNIVERSITY

School of Computer Science, Engineering and Applications

Academic Year 2023-2024(monsoon semester)

Practical Assignment- 08

Subject: Design and Analysis of algorithm

Name: Suryakant Upadhyay

PRN:20220802043

Class:S.Y. B.Tech. SEM III

Topic: Graph

1) Write a C code to implement prims algorithm.

Answer:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <limits.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] && key[v] < min) {
            min = key[v];
            min_index = v;
        }
    }

    return min_index;
}
void 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] && graph[u][v] < key[v]) {
                parent[v] = u;
                key[v] = graph[u][v];
            }
        }
    }

    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}};

    primMST(graph);

    return 0;
}
```

Output:

```
Edge    Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5
```

2) Write a C code to implement Rabin karp algorithm.

Answer:

```
#include <stdio.h>
#include <string.h>
#define d 256

void searchRabinKarp(char pattern[], char text[], int q) {
    int M = strlen(pattern);
    int N = strlen(text);
    int i, j;
    int p = 0;
    int t = 0;
    int h = 1;

    for (i = 0; i < M - 1; i++)
        h = (h * d) % q;

    for (i = 0; i < M; i++) {
        p = (d * p + pattern[i]) % q;
        t = (d * t + text[i]) % q;
    }

    for (i = 0; i <= N - M; i++) {
            for (j = 0; j < M; j++) {
                if (text[i + j] != pattern[j])
                    break;
            }
            if (j == M) {
                printf("Pattern found at index %d\n", i);
            }
        }
        if (i < N - M) {
            t = (d * (t - text[i] * h) + text[i + M]) % q;
            if (t < 0)
                t = (t + q);
        }
    }
}

int main() {
    char text[] = "AABAACAADAABAAABAA";
    char pattern[] = "AABA";
    int q = 101;

    searchRabinKarp(pattern, text, q);

    return 0;
}
```

Output:

```
Pattern found at index 0
Pattern found at index 9
Pattern found at index 13
```