AD Soyad: Basel KELZIYE

Okul No: 20011906

Ders: Algoritma Analizi Gr2

Ders Yürütücüsü: Doç. Dr. Mehmet Amaç GÜVENSAN

YÖNTEM:

Problem:

Verilen bir yönlü grafın her düğümün in_degree değeri verilen bir ‘threshold’ değerinden düşük ise bu düğüm graftan koparılmalıdır. Bir düğümü graftan çıkarınca başka düğümlerin in_degree değerlerini etkilediği için de in-degree değerleri tekrar kontrol edilir, Graftaki her düğüm Sağladığını tespit edene kadar.

Yeni Güncellenmiş Grafla her düğüme giden toplam yol sayısı o düğümün in-degree sayısından büyük ise o düğümü Kullanıcıya bildir.

Çözüm:

In-degree sayısını hesaplamak için bfs algoritması kullanıldı. Başlangıç düğüm için, İlk bulduğu düğümü dönderen metod yazdım(boş graf verilirse bfs

kullanılmasın diye)

```c
int get_root_node(struct graph_node **my_graph, int V)
{
    int i, j;

    for (i = 0; i < V; i++)
    {
        for (j = 0; j < V; j++)
        {
            if (my_graph[j][i].filled == 1)
            {
                return i;
            }
        }
    }
    return -1;
}
```

Düğüm sayısı

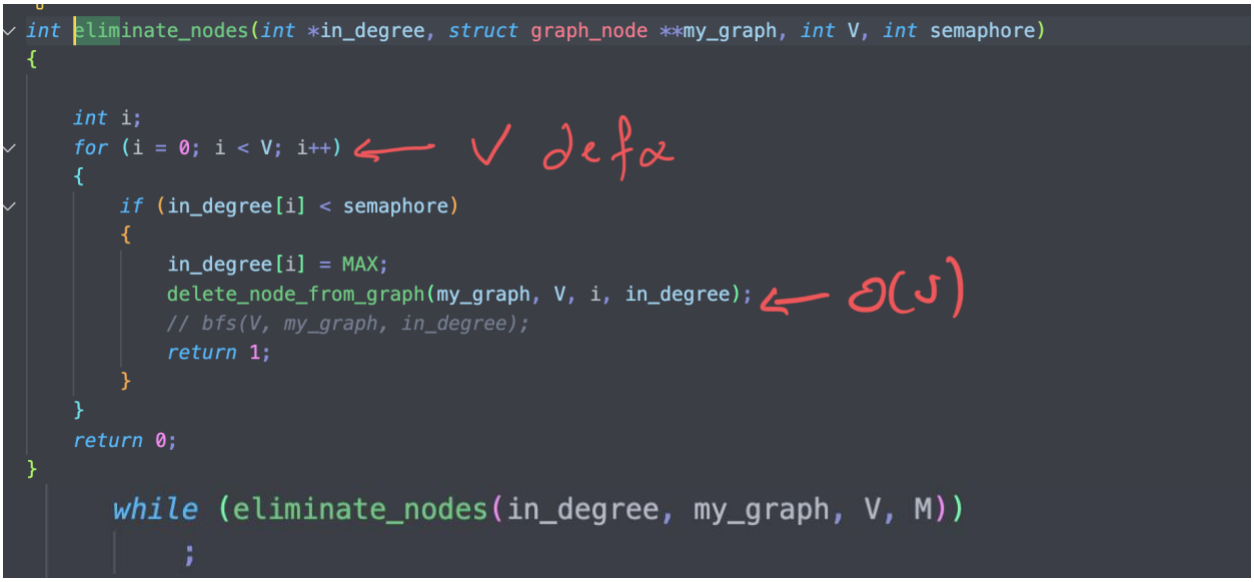
BFS:

```c
int rear = -1, front = -1, i;//kuyruk icin degiskenler
int root_node = get_root_node(my_graph, V);
int *color = (int *)calloc(V, sizeof(int)); // bfs icin renkler dizisi
if (root_node == -1)
{
    printf("\nERROR! Unable to get root node!");
    exit(101);
}
printf("\nStarting Node is : %d", root_node);
enqueue(queue, &front, &rear, root_node, V);// kuyruga buldugumuz root node i at
do
{
    int u = front_queue(queue, rear, front);
    dequeue(queue, &rear, &front, V);
    for (i = 0; i < V; i++)
    {
        if (my_graph[u][i].filled == 1) ← Bağlantı var ise.
        {
            in_degree[i]++; Varış Düğümün in_degreesini arttır.
            if (color[i] == WHITE) → önceden gezilmemişse.
            {
                // printf("\nFrom Node %d to Node %d", u + 1, i + 1);
                enqueue(queue, &front, &rear, i, V);
                color[i] = GREY; ←keşfedilmiş olarak işaretle
            }
        }
    }
    color[u] = BLACK; Düğümün bütün bağlantılarını keşfettik.

} while (!isEmpty(rear, front));
```

Sonra Bütün Yolları Hesaplamak için Grafın Bütün Yönlerini değiştirdim. Her düğümden diğer düğümlere kaç yolla gidilir diye hesaplattım.

C) sonuç:



While da çağrıldığı için (max V defa) Toplam karmaşıklık V^3 olur.

Bfs nin karmaşıklığı komşuluk matrisi olduğu için V^2 dir.

```
    // counter++;
    enqueue(queue, &front, &rear, node, V);
    do
    {
        int u = front_queue(queue, rear, front);
        dequeue(queue, &rear, &front, V);
        for (i = 0; i < V; i++)
        {
            if (my_graph[u][i].filled == 1)
            {

                if (color[i] == WHITE)
                {
                    // printf("\nFrom Node %d to Node %d", u + 1, i + 1);
                    counter++;
                    enqueue(queue, &front, &rear, i, V);
                    color[i] = GREY;
                }
            }
        }
        color[u] = BLACK;
```

Count Fonkisoynu V defa çağrılıyor (her düğüm için) ve her çağrılmada BFS Uyguluyor (karmaşıklık V^2) her defa yaptigi için de Toplam karmasiklik V^3 oldu.

(Video da Test caseler denenmiştir, o yüzden çıktı koymadım raporda.)

https://www.youtube.com/watch?v=DZA2nNC5lJQ