Fatma Aydoğdu 19253061

Cansu Dal 18253039

Problem

Bir bölgedeki trafik sıkışıklığının azaltılması amacıyla verimli bir trafik yönetim sistemi geliştirilmesi gösterilebilir. Bu sistemi oluşturmak için, kavşaklardaki trafik ışıklarının nasıl koordine edileceği ve hangi yolların öncelikli olarak kullanılacağı gibi konuların belirlenmesi gerekmektedir.

Çözüm

Bu problem için, verilen bölgedeki yollar ve kavşaklar bir graf olarak modellenebilir. Aşağıdaki graf, örneğin bir bölgedeki 6 kavşağı ve bunları birbirine bağlayan yolları göstermektedir:

A---B

/| |\

E | | F

\| |/

D---C

Bu grafa göre, öncelikle bağımsızlık sayısı hesaplanabilir. Bir grafın bağımsızlık sayısı, o grafın herhangi bir bağımsız kümesinin eleman sayısıdır. Bu graf için, herhangi bir bağımsız küme 2 elemandan fazla olamayacağından, bağımsızlık sayısı 1'dir.

Daha sonra örtü sayısı veya baskınlık sayısı hesaplanabilir. Ancak bu grafın örtü sayısı ve baskınlık sayısı 2'den büyük olduğu için, bunların hesaplanması zor olacaktır. Bunun yerine, kromatik sayı hesaplanarak trafik ışıklarının koordinasyonu belirlenebilir.

Kromatik sayı, bir grafın düğümlerinin en az kaç renkle boyanması gerektiğini ifade eder, böylece aynı renge boyanmış iki düğüm birbirine bitişiktir. Bu grafın kromatik sayısı 3'tür, yani en az 3 renk gereklidir. Bu nedenle, trafik ışıkları 3 farklı renkle (örneğin kırmızı, sarı ve yeşil) koordine edilebilir.

Trafik ışıklarının nasıl koordine edileceği ile ilgili olarak, kromatik sayı çözümünün birden fazla olabileceği unutulmamalıdır. Örneğin, yukarıdaki grafda, A-B-C-D yolunun ışıkları aynı anda yeşil veya kırmızı olabilir, ancak her durumda E-F yolu için yeşil ışık açılmalıdır. Benzer şekilde, E-D-C-B yolunun ışıkları aynı anda yeşil veya kırmızı olabilir, ancak her durumda A-F yolu için yeşil ışık açılmalıdır.

Problemin Python ile Çözümü

def kromatik\_sayi(graph):

# düğümleri renklere atamak için kullanılacak bir sözlük oluşturulur

colors = {}

# düğümleri büyükten küçüğe sıralamak için bir liste oluşturulur

nodes = sorted(graph, key=lambda x: len(graph[x]), reverse=True)

# tüm düğümleri beyaz renkle başlatırız

for node in nodes:

colors[node] = None

# her düğüm için renk atanır

for node in nodes:

if colors[node] is None:

# düğümün etrafındaki tüm düğümleri kontrol ederek kullanılmayan renkler belirlenir

used\_colors = set([colors[n] for n in graph[node] if colors[n] is not None])

available\_colors = set(range(len(nodes))) - used\_colors

# düğüme kullanılmayan bir renk atanır

colors[node] = min(available\_colors)

# en yüksek renk sayısı kromatik sayı olarak döndürülür

return max(colors.values()) + 1

# örnek bir graf için kromatik sayı hesaplanır

graph = {

'A': set(['B', 'D', 'E']),

'B': set(['A', 'C', 'E']),

'C': set(['B', 'D', 'F']),

'D': set(['A', 'C', 'E']),

'E': set(['A', 'B', 'D', 'F']),

'F': set(['C', 'E'])

}

print(kromatik\_sayi(graph)) # 3