+ALGORİTMALARIN SINIFLANDIRILMASI

Algoritmalar, problemlerin çözümü için uyguladıkları yönteme göre sınıflandırılabilir.

Algoritmaları sınıflandırmadaki temel amaç, problemlerin çözümünde başvurulabilecek

değişik metotları ve alternatifleri tespit edebilmektir.

• Özyinelemeli Algoritmalar (Simple Recursive Algorithms)

• Geri İzlemeli Algoritmalar (Backtracking Algorithms)

• Böl ve Yönet Algoritmaları (Divide and Conquer Algorithms)

• Dinamik Programlama (Dynamic Programming)

• Açgözlü Algoritmalar (Greedy Algorithms)

• Kaba Kuvvet Algoritmaları (Brute Force Algorithms)

-Özyinelemeli Algoritmalar (Simple Recursive Algorithms)

Kendisini doğrudan veya dolaylı olarak çağıran algoritmalara özyinelemeli algoritma adı

verilir. Bu algoritmalarda, problemler daha küçük ve basit parçalara indirgenir. Küçük

parçalar için oluşturulan çözümlerin birleştirilmesiyle ana problemin çözümü elde edilir.

Faktöriyel hesabı, özyinelemeli bir algoritma kullanılarak çözülebilecek problemlere

güzel bir örnektir. 5 sayısının faktöriyeli bulunmak istendiğinde, 5’ten 1’e kadar olan tam-

sayılar çarpılır. Bu problemin özyinelemeli bir algoritma ile çözümünde aşağıdaki adımlar

uygulanır:

• Algoritmanın çıkış koşulu belirlenir (1! = 1).

• 2 sayısının faktöriyeli hesaplanır (2! = 1! \* 2 = 2).

• 3 sayısının faktöriyeli hesaplanır (3! = 2! \* 3 = 6).

• 4 sayısının faktöriyeli hesaplanır (4! = 3! \* 4 = 24).

• 5 sayısının faktöriyeli hesaplanır (5! = 4! \* 5 = 120).

• Beklenen hesaplamaya ulaşıldığı için algoritma sonlandırılır.

-Geri İzlemeli Algoritmalar (Backtracking Algorithms)

Geri izlemeli algoritmalar, genellikle optimizasyon problemlerinde kullanılan, prob-

lem çözümünde tüm olasılıkları deneyen algoritmalardır. Bu algoritmalarda çözüm

kademeli şekilde oluşturulur. Algoritma çözüm aşamasında ilerlerken, olası çözüm

yollarının hepsini deneyerek bir sonraki adıma geçmeye çalışır. Algoritmanın dene-

diği çözüm yolundan sonuç alınamazsa, algoritma bir önceki adımda bulunan diğer

olası çözüm yollarına geri döner.

-Böl ve Yönet Algoritmaları (Divide and Conquer Algorithms)

Böl ve yönet algoritmaları, problemlerin mümkün olan en küçük alt parçalara ayrıldığı,

her bir alt parçanın diğerlerinden bağımsız şekilde çözüldüğü algoritmalardır. Problemin

genel çözümü elde edilirken alt parçalara ait çözümler belirli bir sırayla bir araya getirilir.

Böl ve yönet algoritmaları, genellikle üç ana aşamadan meydana gelmektedir:

• Bölme (Divide): Problemin daha küçük parçalara ayrıldığı aşamadır. Problem

daha alt parçalara bölünemeyecek hale gelene kadar, özyinelemeli bir yaklaşımla

bölme işlemi gerçekleştirilir.

• Yönetme (Conquer): Problemin alt parçalarının, birbirlerinden bağımsız olarak

çözüldüğü aşamadır.

• Birleştirme (Merge): Problemin alt parçalarına ait çözümlerin, özyinelemeli bir

yaklaşımla birleştirildiği aşamadır.

-Dinamik Programlama (Dynamic Programming)

Dinamik programlama, karmaşık problemleri küçük parçalar halinde çözen, elde edilen

sonuçları bilgisayar hafızasında bir veri yapısında saklayan, genel çözümü elde ederken de

veri yapılarında saklanan sonuçları kullanan bir programlama yöntemidir.

Bir problemin dinamik programlama ile çözülebilmesi için problemin alt parçalara

ayrılabilmesi ve genel çözümün bu alt parçalardan oluşturulabilmesi gerekmektedir. Di-

namik programlama yaygın olarak optimizasyon problemlerinde kullanılır.

-Açgözlü Algoritmalar (Greedy Algorithms)

Bir problem için mümkün olan en doğru çözümü hedefleyen algoritmalara açgözlü algo-

ritmalar adı verilir. Açgözlü algoritmalarda yerel olarak optimum sonuç elde edilirken,

bulunan sonuç her zaman için en iyi çözüme karşılık gelmeyebilir.

Açgözlü algoritmalar ile problem çözümündeki temel yaklaşım, problemin küçük bir

alt kümesi için çözüm oluşturmak ve bu çözümü problemin geneline yaymaktır. Algorit-

ma içerisinde yapılan bir seçim, o an için doğru olsa bile sonraki seçimlerde olumsuz etki

yapabilir.

-Kaba Kuvvet Algoritmaları (Brute Force Algorithms)

Bir problemin çözümü aşamasında, kabul edilebilir bir çözüm elde edene kadar tüm ola-

sılıkları deneyen algoritmalara kaba kuvvet algoritmaları denir.

Kaba kuvvet algoritmaları, genellikle problemin tanımından yola çıkarak en basit çö-

züm yolunu uygular ve rahatlıkla kodlanır. Fakat bu algoritmalarda çok fazla işlem yapılır

ve çözüm yolu optimumdan uzaktır. Problemdeki veri hacmi büyüdükçe, kaba kuvvet

algoritması ile çözüm şansı da azalır.

Bir liste içerisinde eleman aramak, kaba kuvvet algoritmaların kullanımıyla çözülebi-

lecek problemlere bir örnektir. Listenin tüm elemanları sırayla kontrol edilerek, aranan

elemanın listede olup olmadığına bakılabilir. Listenin eleman sayısı arttıkça, kaba kuvvet

algoritmasının çalışma süresi ve yaptığı karşılaştırmalar da artacaktır.