# ALGORİTMAYI NEDEN ANALİZ EDERİZ?

Günlük hayatımızda birçok işimizi en kısa yoldan yapmaya çalışırız. Programlamada da durum böyledir. Büyük bir veriyi işlerken olabildiğince hızlı yapılması gerekir. Buradaki hızlı ve yavaş kavramları, girdi boyutu ile doğrudan ilişkilidir. Örneğin 1 tane veriyi 100 tane veriye göre çok daha hızlı sıralayabiliriz.

Yani girdi miktarını örneklendirirerek kıyas yaparız ve bir tahminde bulunuruz. Çünkü işlenmesi gereken veri miktarı çoğu zaman belli değildir. Bu yüzden, emin olamadığımız sayıda veri girişi içeren bir **algoritmayı tasarlamak** veya tercih etmek için analize ihtiyaç duyarız.

Programlamada yapılan işin kalitesi açısından önemlidir. Problemlerin çözümü için rastgele çözüm fikirleri oluşturmak ya da rastgele algoritmalar yaratmak programın kalitesini olumsuz etkiler. Problemlere özel olan en uygun çözüm algoritmasını belirlemek problemi çözmektenden daha önemli bir adımdır.

Çünkü günün sonunda algoritma bir şekilde oluşur ve sorun çözülür ama her bakımdan maksimum performansı elde eden tek bir algoritma vardır.

İşte o **algoritmayı bulmak**, diğer algoritmalarla arasındaki farkları ve avantajları belirlemek veya kısaca **kıyas yapmak için algoritma analizine** ihtiyaç duyarız. Bazen de karşılaştırma yapmaktan ziyade sadece bir **algoritmanın performansını ölçmek** için analize ihtiyaç duyarız.

# ALGORİTMA ANALİZİNİ NASIL YAPARIZ?

Algoritmanın zaman karmaşıklığını hesaplamak o algoritmada operasyon sayısını saymaktır. Bu zaman karmaşıklığı derleyiciden bağımsız olmalıdır.

Algoritma analizinde genellikle karmaşıklık teoremi kullanılmaktadır. Matematiksel hesaplamalar ile algoritma performansı hakkında bilgiler edinilebilir, yapılan analiz bir algoritmanın asimtotik değerlerini bulmayı hedefler.

Yaygın olarak kullanılan üç değer:

Big OH Notasyonu – Zaman karmaşıklığında üst sınırı gösterir. En kötü durum analizidir. Big-O notation bir algoritmanın performansını veya time complexity’sini hesaplamak için kullanılır.

Time complexity - Bir algoritmanın çalışması için gerekli olan süredir. Ancak buradaki süre, saniyeleri hesaplayarak değil, kaç tane işlem gerçekleştirdiğine göre hesaplanmaktadır. Uygulama tarafından gerçekleştirilen işlem sayısı, veri setinin büyüklüğüne ve o veri setindeki elemanlarına sırasına göre belirlenir.

Time complexity bize bir algoritma için 3 durum sunar. **Best-case**, **average-case** ve **worst-case.**

Best Case(En iyi durum)- Bulunan ifadeden daha hızlı algoritma çalıştırılamaz

Average-case (Ortalama)- Ortalama bir değeri ifade eder.

Worst-case(En kötü durum)- Bulanan ifadeden daha yavaş bir algoritma çalıştırılamaz

Big Omega Notasyonu Big Oh notasyonunun tam tersidir. Zaman karmaşıklında alt sınırı gösterir.

Big Theta Notasyonu Bu notasyon big Oh notasyonu ile big Omega notasyonu arasında ortalama bir karmaşıklığı ifade eder.