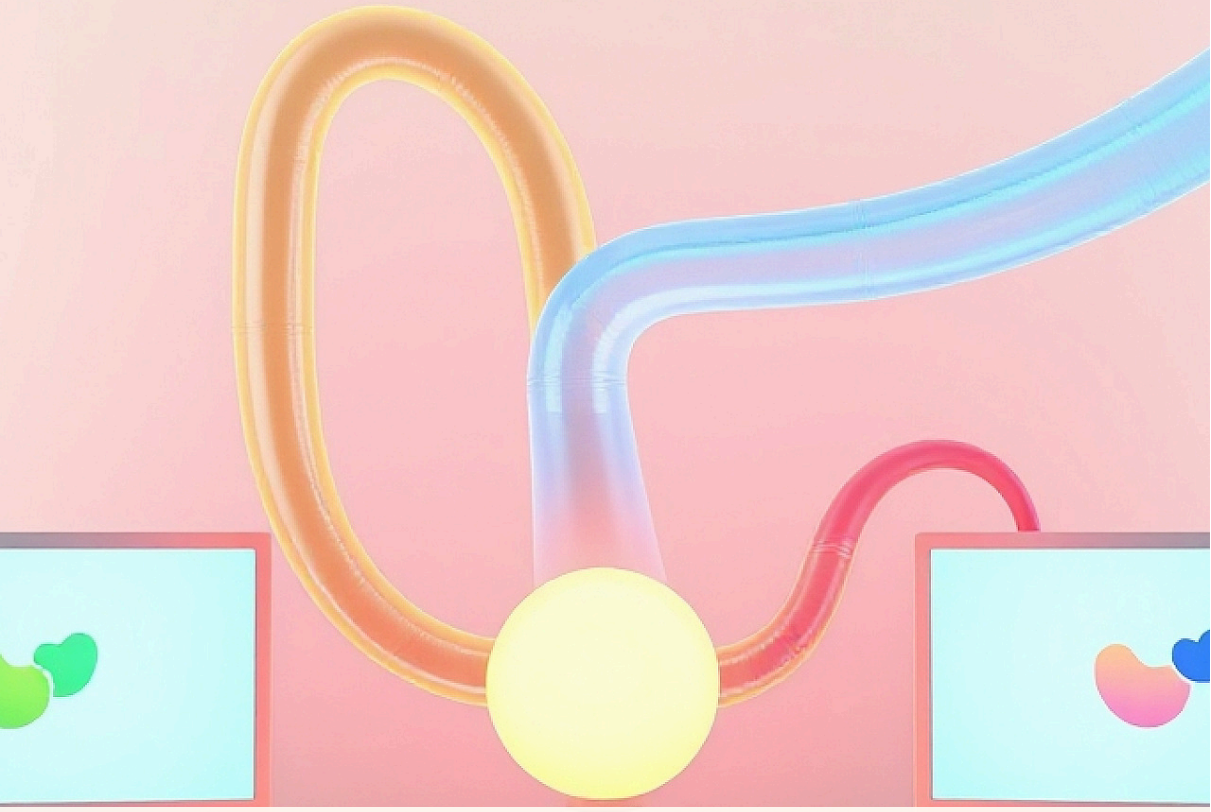
# Aynı işi yapan iki program neden farklı hızlarda çalışır?



www.yzokul.com.tr

# Aynı işi yapan iki program neden farklı hızlarda çalışır?



İki program aynı çıktıyı üretir, yine de biri bir saniyede bitirirken diğeri bir dakika sürer. Kısa yanıt, hızın dört şeye bağlı olduğudur: algoritma, veri yapısı, yürütme sistemi ve donanım. Bu kitap her birini sıfırdan, takip edebileceğin küçük örneklerle açıklar. Veri büyüdükçe farkın neden arttığını göreceksin.



Bir program, girdi alıp çıktı üreten bir adımlar listesidir. Örneğin, bir isim listesi gir, en uzun ismi çıkar. İki program aynı çıktıyı farklı adım listeleriyle üretebilir. Bir liste her ismi iki kez kontrol edebilir; bir başkası her ismi bir kez kontrol eder. Çıktı aynıdır, ama adım sayısı aynı değildir.



Bir programın kaç temel işlem yaptığını say: karşılaştırmalar, toplamalar, bellek okumaları. Daha çok işlem genellikle daha çok zaman demektir. İki programı adil biçimde karşılaştırmak için aynı girdi boyutunda adımlarını say. Örneğin, on isimlik bir listede bir ismi bulmak bir programda on karşılaştırma, diğerinde beş karşılaştırma sürebilir. Adım saymak, hızı karşılaştırmanın ilk yoludur.



Algoritma, bir programın bir problemi çözmek için izlediği adımlar tarifidir. Aynı yemeğin iki tarifi, gerektirdikleri iş miktarı bakımından farklı olabilir. Bir tarif tenceredeki yemeği elli kez karıştırabilir; bir başkası on kez karıştırıp aynı sonucu alır. Programlarda daha iyi bir algoritma, yanıta daha az işlemle ulaşır, bu yüzden aynı makinede daha hızlı çalışır.



Bir program aynı öğeleri gereksiz yere tekrar tekrar kontrol edebilir. Örneğin, en büyük sayıyı bulmak için her sayı çiftini defalarca karşılaştırır. Bir başka program listeyi bir kez tarar ve o ana kadar gördüğü en büyüğü tutar. İkisi de aynı yanıtı verir, ama ikincisi tekrarlanan işi atlar. Daha az işlem daha az zaman demektir, özellikle liste uzunken.



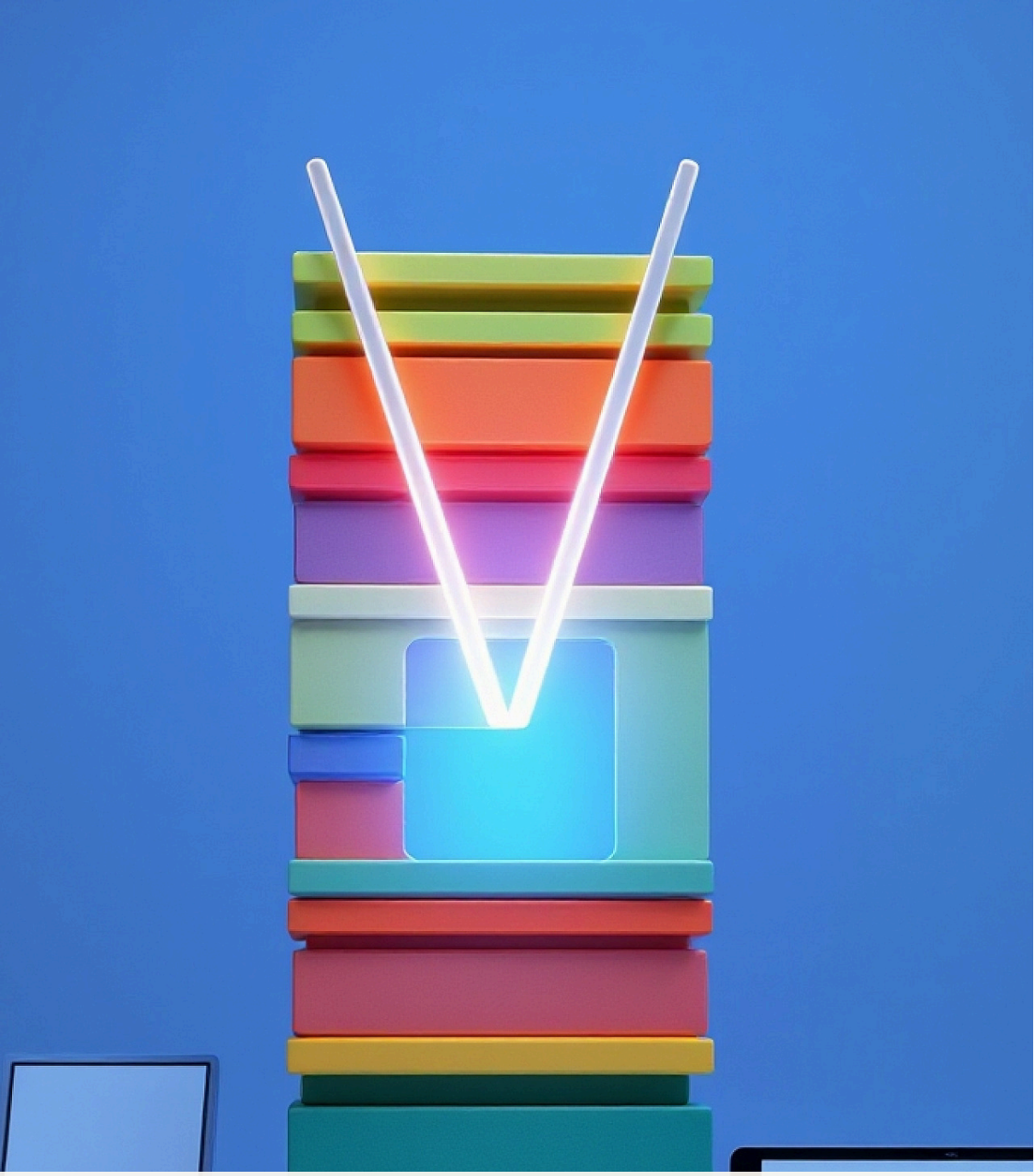
Veri yapısı, öğelerin bellekte nasıl düzenlendiğidir. Aynı algoritma bu düzenlemeye bağlı olarak hızlı ya da yavaş olabilir. Örneğin, isimler taraman gereken düz bir listede saklanıyorsa bir ismi bulmak yavaştır. İsimler sıralı bir listede ya da bir karma tabloda saklanıyorsa program yanıta daha çok yaklaşabilir. Doğru yapıyı seçmek birçok işlemi kurtarır.



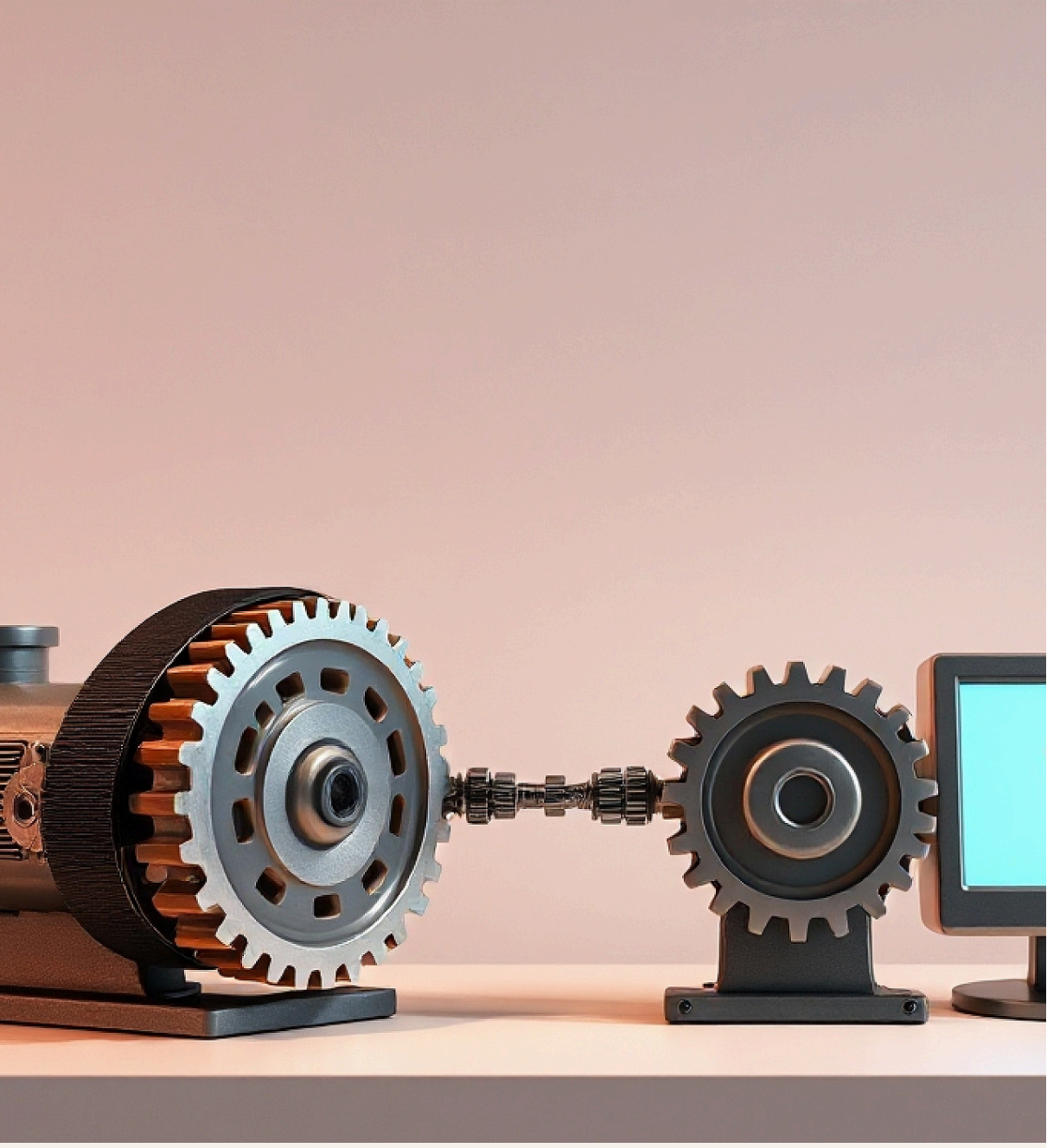
Basit bir arama, hedefi bulana kadar öğeleri tek tek kontrol eder. Örneğin, on sayılık bir listede 7 sayısını bulmak için onunun hepsini kontrol edebilirsin. Bu işe yarar, ama kontrol sayısı listenin boyutuyla büyür. Listede bin öğe varsa bin kontrol gerekebilir. Bu, büyük veriler için yavaştır.



Veri sıralı olduğunda, uygun bir arama yöntemi işi çok azaltabilir. Sıralılık, programın her öğeyi kontrol etmek yerine bütün bölümleri atlamasını sağlar. Örneğin, sıralı bir sayı listesinde ortadaki sayı çok küçükse hedefin sol yarıda olmadığını bilirsin. O yarıyı tamamen yok sayabilirsin. Sıralı veri, daha hızlı yöntemlere olanak tanır.



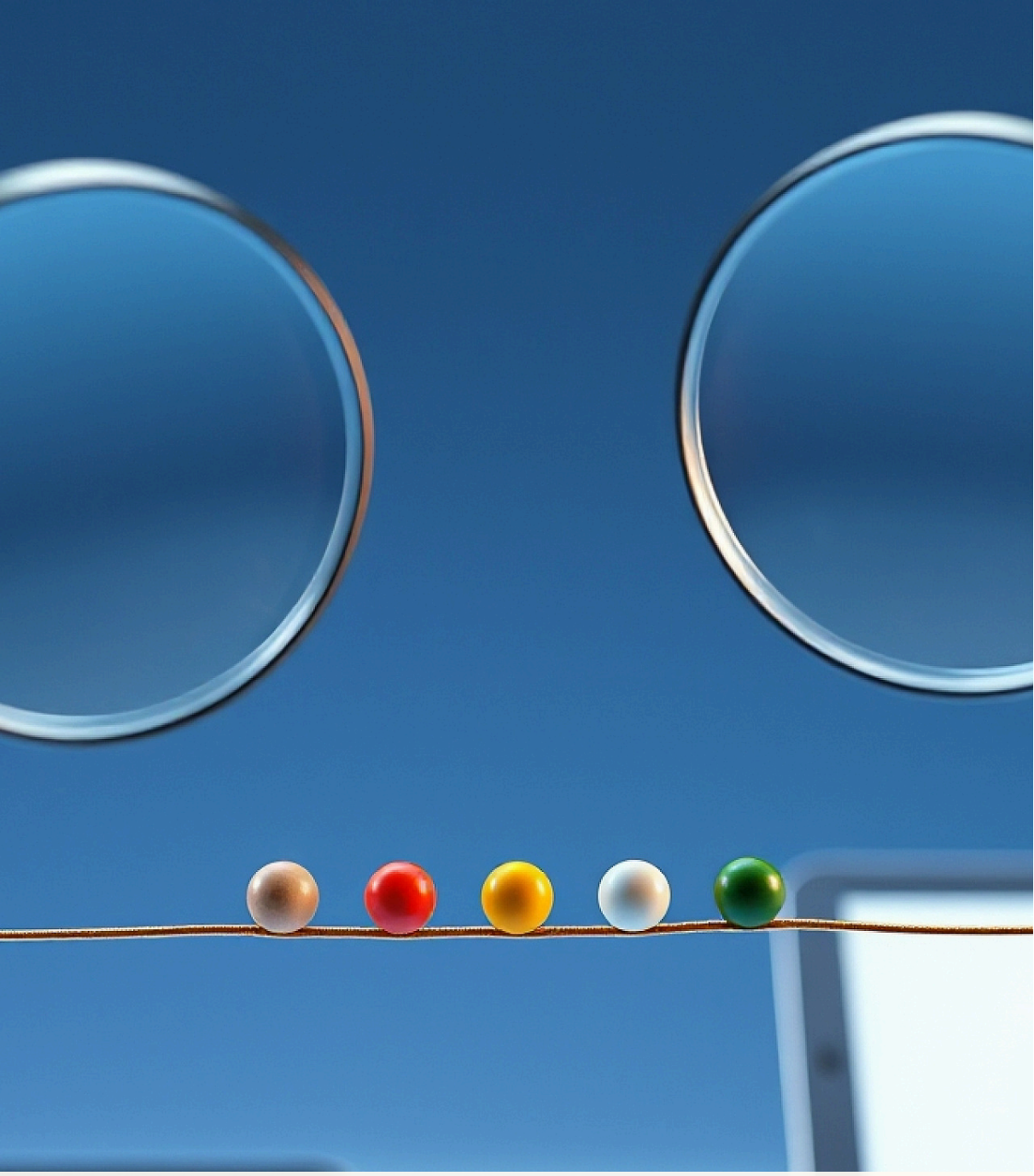
Orta öğeye bakıp her seferinde listenin yarısını atan bir arama çok daha az adım gerektirir. Örneğin, bin sayılık sıralı bir listede bin değil, yaklaşık on kontrol yeter. Her kontrol kalan öğeleri yarıya indirir: 1000, 500, 250 ve böyle devam eder. Buna ikili arama denir ve iyi bir arama yönteminin sıralı verilerde nasıl yardımcı olduğunu gösterir.



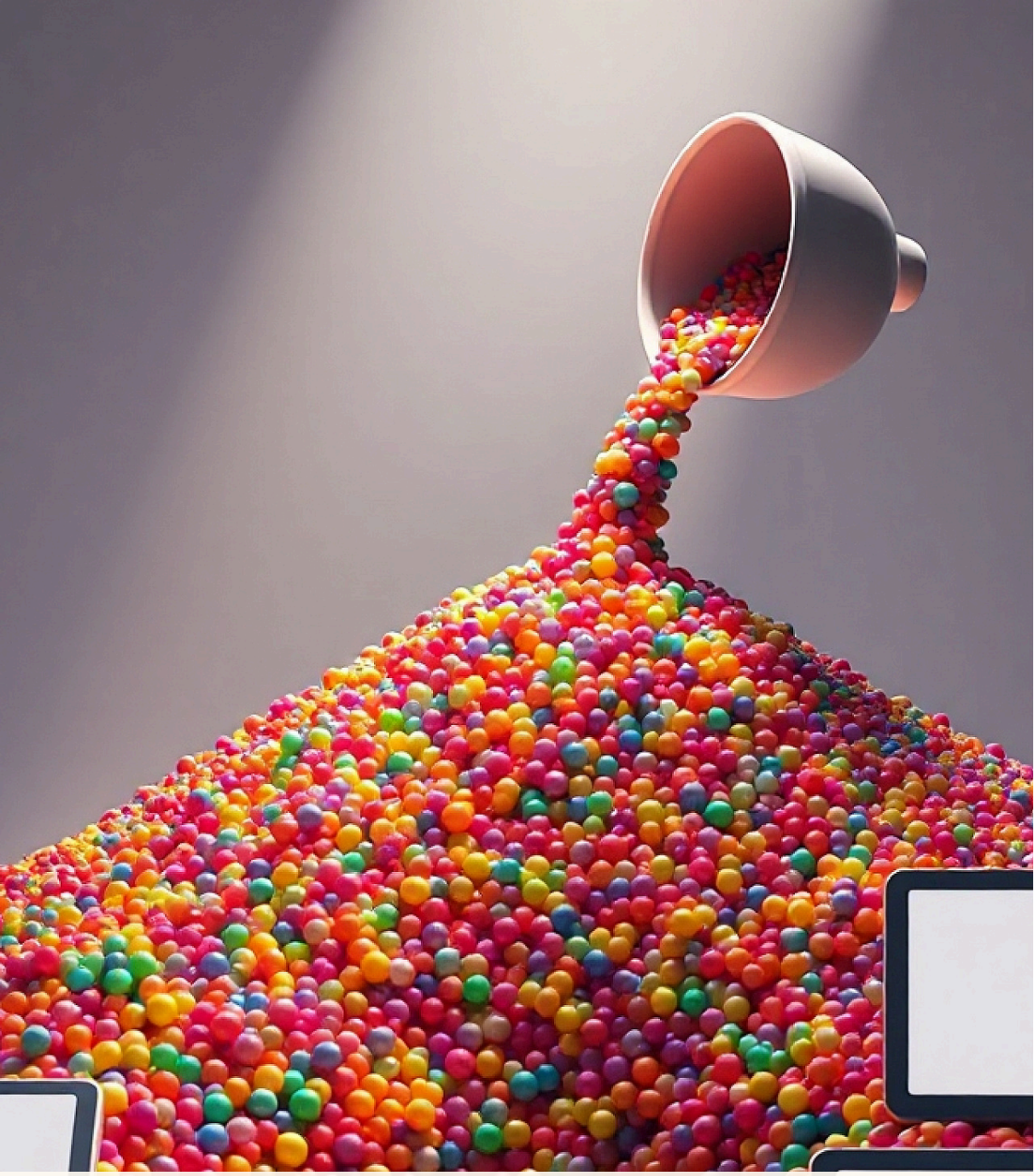
Bir dilin yürütme sistemi hızı etkiler. Bazı kodlar önceden makine komutlarına çevrilir, böylece doğrudan işlemcide çalışır. Bazı kodlar ise çalışırken satır satır yorumlanır, bu da ek yük getirir. Bu yüzden aynı adımlar, dilin onları nasıl yürüttüğüne bağlı olarak daha hızlı ya da daha yavaş çalışabilir. Benzer mantığa sahip iki programın farklı olmasının bir nedeni budur.



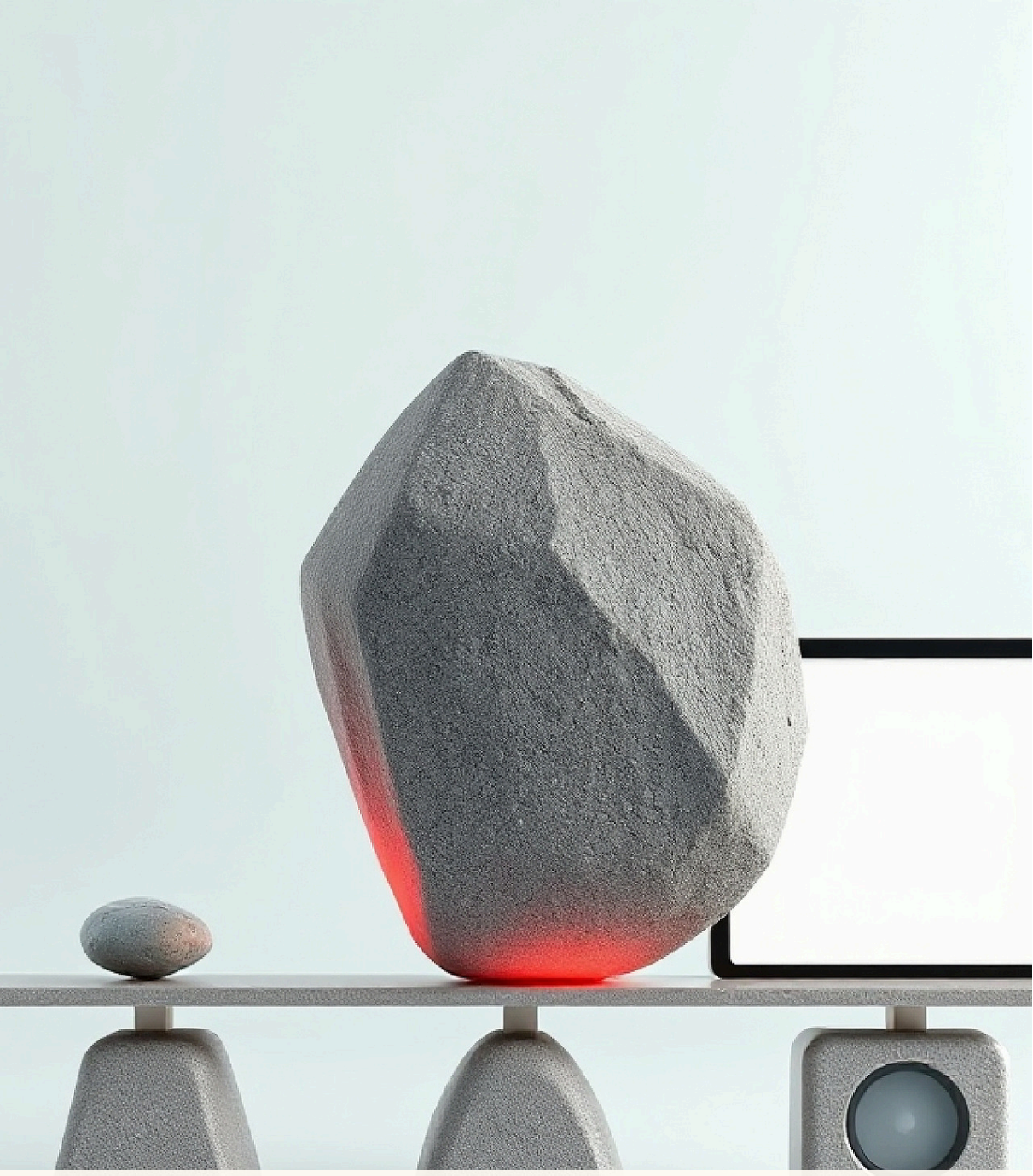
Aynı program farklı makinelerde farklı hızlarda çalışır. İşlemci hızı, bellek boyutu ve bellek hızı, herhangi bir programın ne kadar hızlı gidebileceğine sınır koyar. Hızlı bir işlemci saniyede daha çok işlem yapabilir; daha çok bellek, yavaşlama olmadan daha çok veri tutabilir. Donanım algoritmayı değiştirmez, ama her işlemin aldığı süreyi değiştirir.



Küçük bir listede iki program neredeyse aynı anda biter. Az öğe varken boşa harcanan işlemler fark edilemeyecek kadar azdır. Örneğin, beş isimlik bir listede her ismi iki kez kontrol etmek yalnızca on kontrol sürer, bu da anlıktır. Yavaş program da aynı hızlı görünür. Küçük örnekler farkı gizler çünkü toplam iş azdır.



Veriyi büyüt ve yavaş programın çok geride kaldığını gör. Örneğin, bir milyon isimlik bir listede her ismi iki kez kontrol etmek iki milyon kontrol sürerken bir kez kontrol etmek bir milyon sürer. Fark belirginleşir. Veri büyüdükçe gereksiz işlemlerin maliyeti açığa çıkar. Bu yüzden gerçekçi boyutlarla test etmelisin.



Hızı tek bir küçük testle yargılama. Küçük veride iyi görünen bir program büyük veride çok yavaş olabilir. Örneğin, her öğeyi kontrol eden bir arama on öğede anlık görünür ama on milyon öğede sonsuza kadar sürer. Her zaman gerçekte kullanacağın boyutlara yakın veri boyutlarıyla test et. Tek bir küçük test seni tamamen yanıltabilir.



Bazı yöntemler önce veriyi sıralamak gibi hazırlık gerektirir. Bu hazırlık zaman alır, bu yüzden programları karşılaştırırken onu da saymalısın. Örneğin, ikili arama hızlıdır, ama ancak liste sıralandıktan sonra, büyük bir listeyi sıralamak da birçok işlem sürer. Yalnızca bir kez arama yapacaksan hazırlık, kazancı aşabilir. Hazırlık ile aramayı birlikte say.



Hız algoritmadan, veri yapısından, yürütme sisteminden ve donanımdan gelir. Veri büyüdükçe en çok daha az gereksiz işlem ve uygun yöntemler önemlidir. Sıralı veride iyi bir arama yöntemi bir milyon kontrolü yaklaşık yirmiye indirebilir. Küçük testler farkı gizler; büyük veri açığa çıkarır. Programları karşılaştırırken bu dört etkeni hatırla.