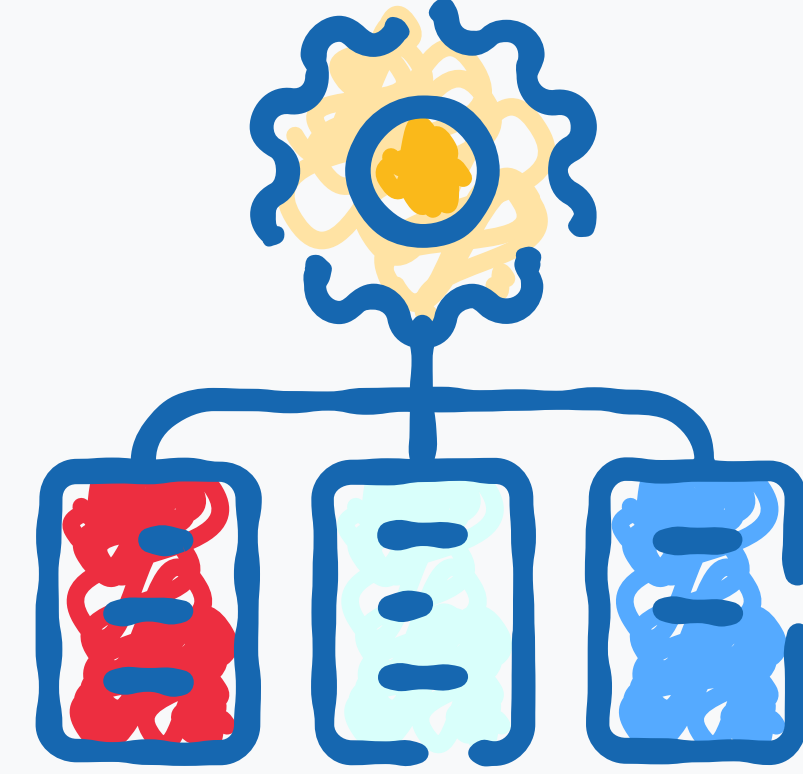


ROBOTİK SİSTEMLERDE ALGORİTMAYI TEST EDEBİLME

Bu Hafta Ne İşleyeceğiz?

Geçen hafta robotlarımıza ne yapacaklarını adım adım anlatan algoritmalar yazdık ve akış şemaları çizdik. Ancak profesyonel yazılımcıların bildiği çok önemli bir sır vardır: Hiçbir algoritma ilk seferinde kusursuz çalışmaz! Kağıt üzerinde harika görünen bir plan, gerçek verilerle karşılaştığında beklenmedik tepkiler verebilir. Bu hafta, yazdığımız kodların güvenilirliğini ölçmek için sahaya iniyor, testler yapıyor ve sistemdeki gizli "böcekleri" avlıyoruz.



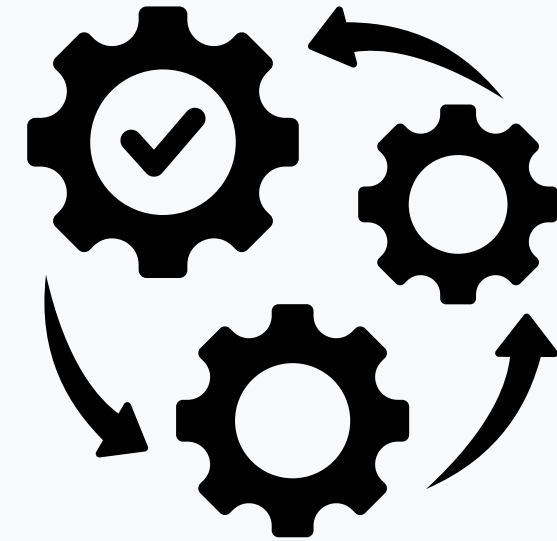
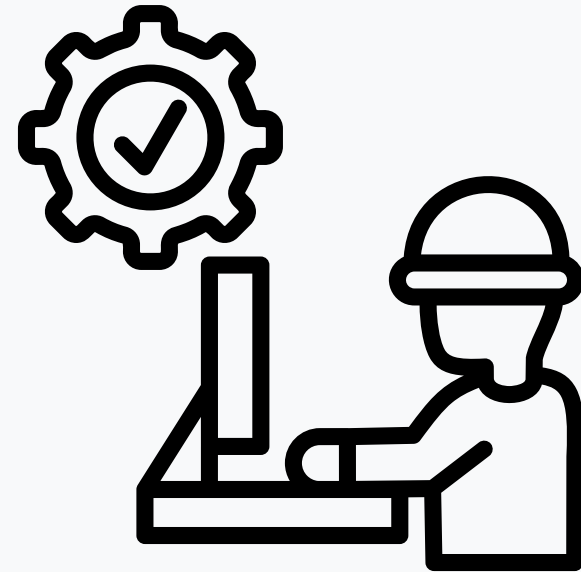
Algoritmayı Neden Test Etmeliyiz?

Test Etmek Ne Demektir?

Tanım: Bir algoritmanın, farklı ihtimaller ve "örnek girdiler" karşısında beklediğimiz doğru çıktıları verip vermediğini kontrol etme sürecidir.

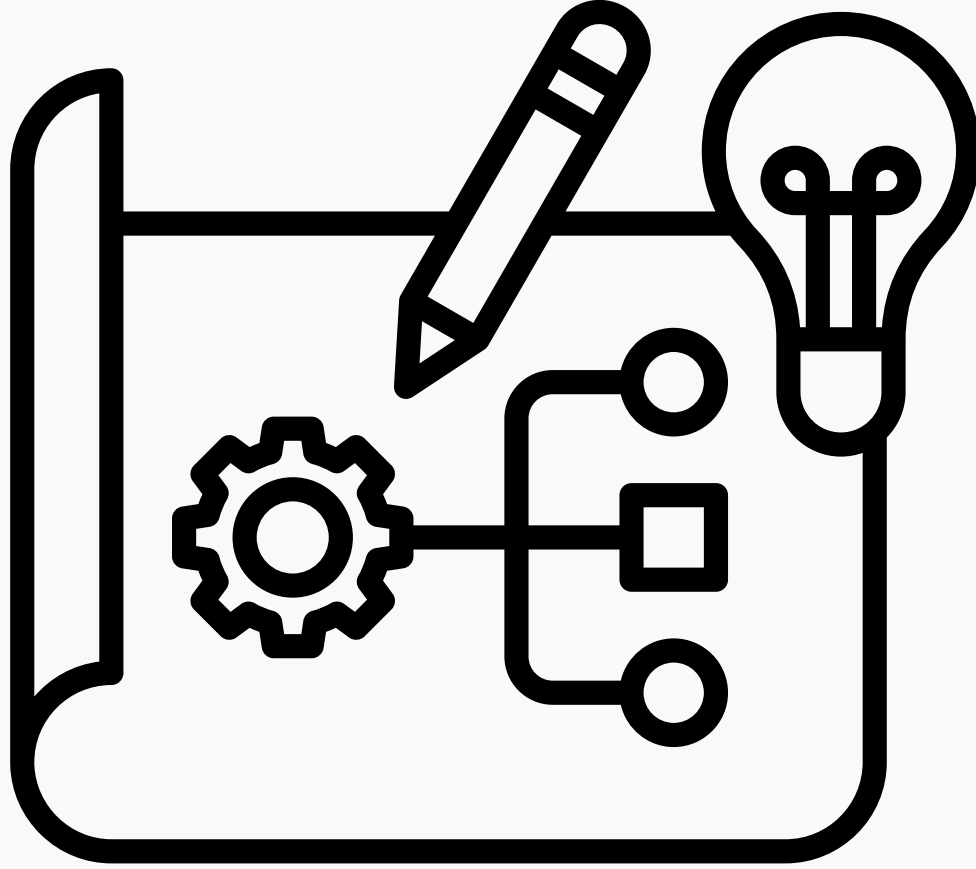
Neden Önemlidir?: Robotlar gerçek dünyada çalışır. Test edilmemiş bir otonom araba kazaya sebep olabilir veya test edilmemiş bir fabrika robotu yanlış ürün paketleyebilir.

Altın Kural: Bir robotik sistemi insanlara sunmadan (kullanıma hazır hâle getirmeden) önce, olabilecek en zor koşullarda test etmeliyiz.



Algoritma Yazarken Nelere Dikkat Edilmeli

KRİTİK NOKTA: Sadece "doğru" verilerle test yapmak yeterli değildir. Örneğin; bir hesap makinesi algoritması yazdınız. $2+2$ yazıp 4 sonucunu gördüğünüzde "Çalışıyor!" diyemezsiniz. Kullanıcı yanlışlıkla sayı yerine bir harf (örneğin $5+A$) girerse sistemin çökmeyip uygun bir uyarı vermesini sağlamak da test sürecinin bir parçasıdır.



Örnek Girdiler ve Örüntüleri Yakalamak

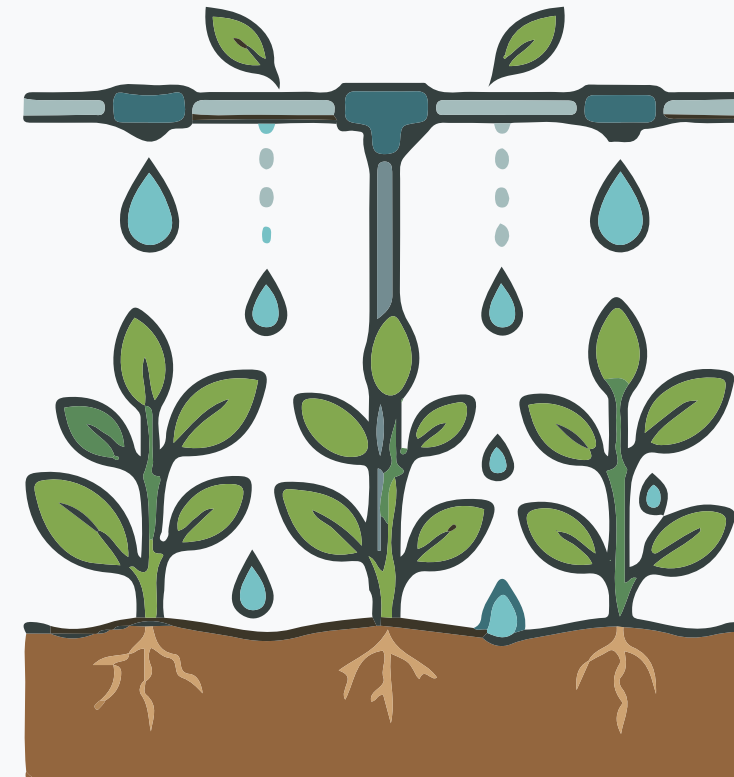
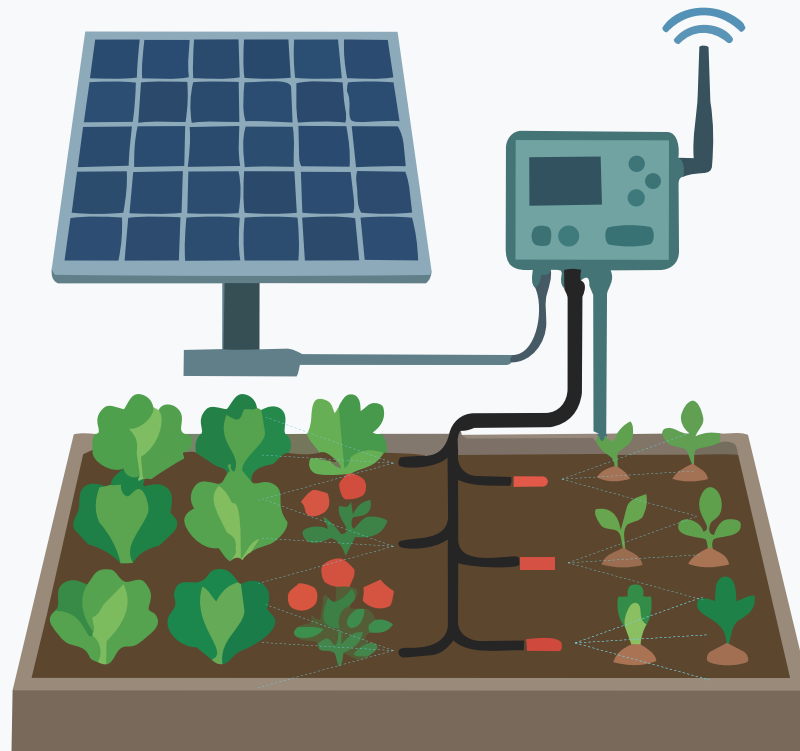
Örnek Girdilerle Robotu Sınamak

Geçen hafta tasarladığımız "Akıllı Sulama Robotunu" hatırlayalım. Şimdi ona hayali örnek girdiler vererek test edeceğiz:

Test 1: Nem %80 (Toprak Islak) → Beklenen Çıktı: Motor Çalışmasın.

Test 2: Nem %10 (Toprak Kuru) → Beklenen Çıktı: Motor Çalışsın.

Test 3: Su Tankı Boş, Nem %5 → Beklenen Çıktı: Kırmızı Işık Yansın, Motor Çalışmasın!



Örnek Girdiler ve Örüntüleri Yakalamak

Verilerdeki Örüntüleri (Şablonları) Anlamak

Diyelim ki robotumuz şu hataları yaptı:

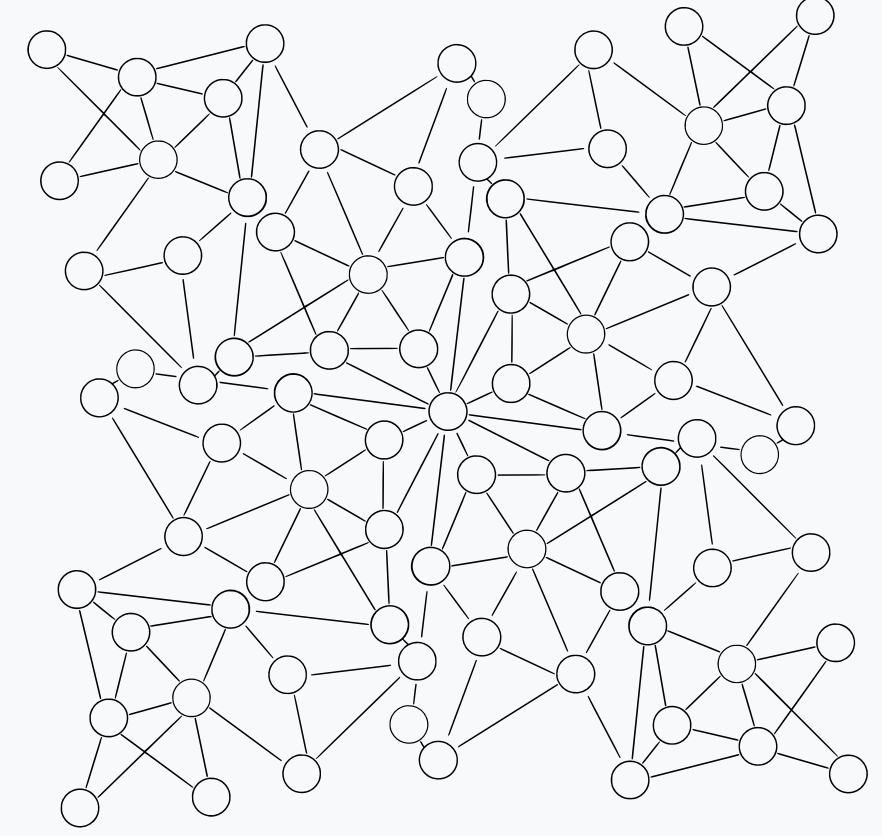
- Nem %30 girildi → Motor çalışmadı.
- Nem %40 girildi → Motor çalışmadı.
- Nem %45 girildi → Motor çalışmadı.



Örnek Girdiler ve Örüntüleri Yakalamak

Örüntü Nedir?

Burada bir örüntü (tekrar eden bir durum) var. Robot, nem oranı %50'nin altına düştüğünde tepki vermiyor. Demek ki algoritmamızdaki "Eğer toprak kuruyorsa" şartını kontrol eden adımda bir mantık hatası var!



Hata yapan bir sistemi düzeltmenin ilk adımı, hatanın ne zaman ve hangi koşullarda tekrar ettiğini bulmaktır. Test verilerindeki bu tekrarlar örüntü denir. Örüntüleri doğru yorumlayan bir bilişimci, sorunun kaynağına çok daha hızlı ulaşır.

Hata Ayıklama (Debugging)

Kodların İçindeki Böcekler (Bug) ve Hata Ayıklama

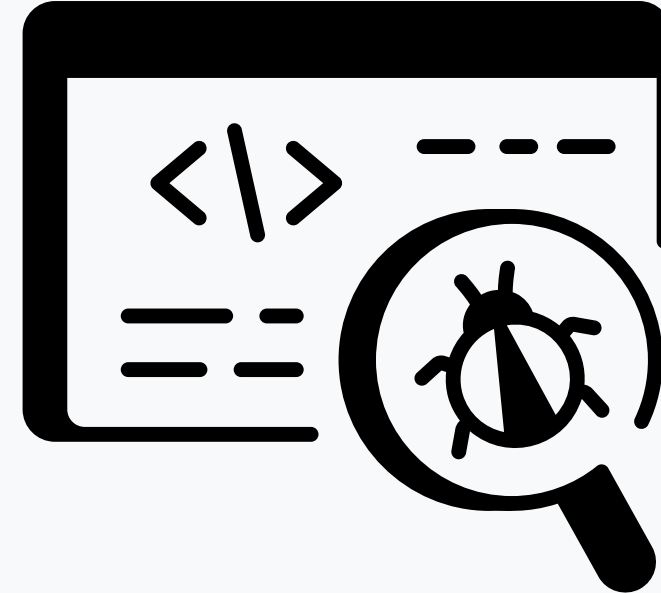


Bug (Böcek) Nedir?

Yazılım ve robotik dünyasında, sistemin yanlış veya eksik çalışmasına neden olan mantık ya da kodlama hatalarına "Bug" denir.

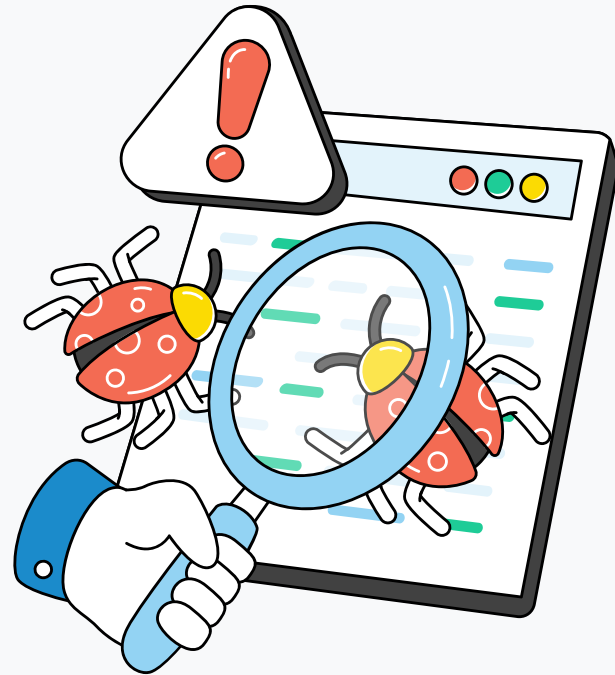
Debugging (Hata Ayıklama) Nedir?

Test aşamasında tespit edilen bu "böcekleri" bulup algoritmamızı düzelterek sistemi kullanıma hazır hâle getirme işlemidir.



Tarihteki İlk Bilgisayar Böceği

Bilişim tarihinde "Bug" kelimesinin nereden geldiğini biliyor musunuz? 1947 yılında ünlü bilgisayar bilimcisi Grace Hopper, devasa büyüklükteki bir bilgisayarın arızalandığını fark eder. İçini açıp baktıklarında, sistemin röleleri arasına sıkışmış gerçek bir güve (böcek) bulurlar! O günden beri bilgisayar hatalarına "bug" diyoruz.



Takip Et! Bul! Düzelt!

Algoritmada hata ayıklamak dedektiflik gibidir. Adımları baştan sona parmağınızla takip eder, robotun her bir adımında ne düşündüğünü (örnek girdilere göre) tahmin eder ve mantığın nerede koptuğunu bulursunuz.



"Kör Uçuş" Uygulaması

Herkes aklından "sınıftan bahçeye gitme algoritması" bir algoritması yazsın.

Algoritmayı yanındaki/önündeki arkadaşına versin. Arkadaşının algoritmasını alan kişi bu algoritmayı satır satır okuyarak parmağıyla kontrol etsin. Eğer algoritma sizi duvara çarpıyorsa ya da çarpma ihtimali görüyorsanız, o adıma "Burada BUG var!" yazın.

Daha sonra birlikte algoritmaları test ederek hataları düzeltin. Kullanılır ve hatasız bir algoritmayı hazır hale getirin.

