Algoritma Mühendisliği

Hesaplama Kuramından Yazılım Mühendisliğine Seyahat

Prof. Dr. M. Oğuzhan Külekci
kulekci@itu.edu.tr
web.itu.edu.tr/kulekci

Teknoloji Transferi Online Seminerleri - PITSTOP Programı

12 Mayıs 2020

M. Oğuzhan Külekci

Kayseri Fen Lisesi'93

Bilgisayar Mühendisliği
Lisans'98

Biyomedikal Mühendisliği
Yüksek Lisans'00

Bilgisayar Bilimleri
Doktora'06

Bilgisayar Mühendisliği
Doktora Sonrası
2009-2010

TÜBİTAK UEKAE

TÜBİTAK MAM

1999 - 2014
Araştırmacı ve idari pozisyonlar

Öğretim üyesi, 2015 -

University of Central Florida

Aalto University

INDIANA UNIVERSITY BLOOMINGTON

Büyük veri yönetimi ve analizine yönelik algoritmalar ve veri yapıları
dizgi eşleştirme (pattern matching), veri sıkıştırma (data compression), algoritmik biyoenformatik, güvenlik ve mahremiyet korunumlu metin işleme….

web.itu.edu.tr/kulekci

Konumuz ...

Teori ile pratiğin arasındaki köprü ....

ALGORİTMA MÜHENDİSLİĞİ

- **Hesaplanabilirlik**
- **Verimli hesaplama**

Yazılım = Algoritma + Veri yapısı

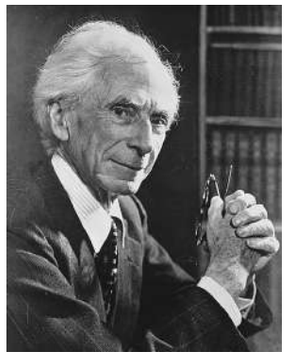
Sunum Planı:

1. **Kuramsal hesaplama tarihinde gezinti**
2. **Kuramsal hesaplamaya hakimiyetin yazılıma katkıları**
3. **Yüksek performanslı yazılım geliştirme için algoritma mühendisliği**

Hesaplanabilirliğin Temelleri 1870 - 1940

Georg Cantor
1845-1917

Küme teorisi
1874

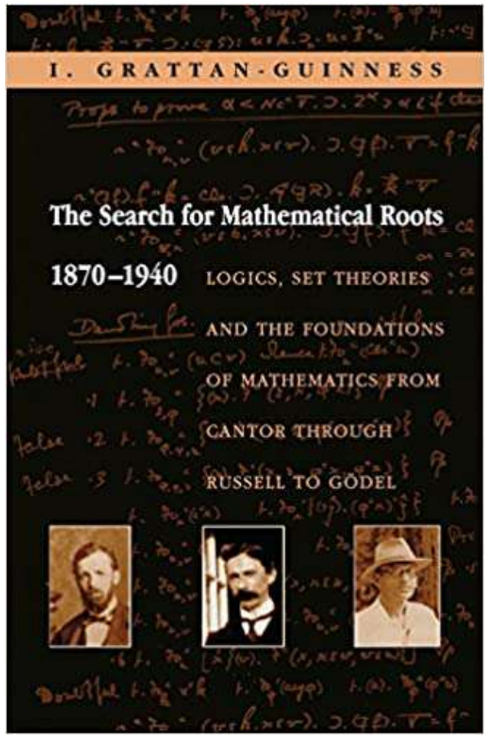


Bertrand Russell
1872-1970

Russel Paradox!
1901

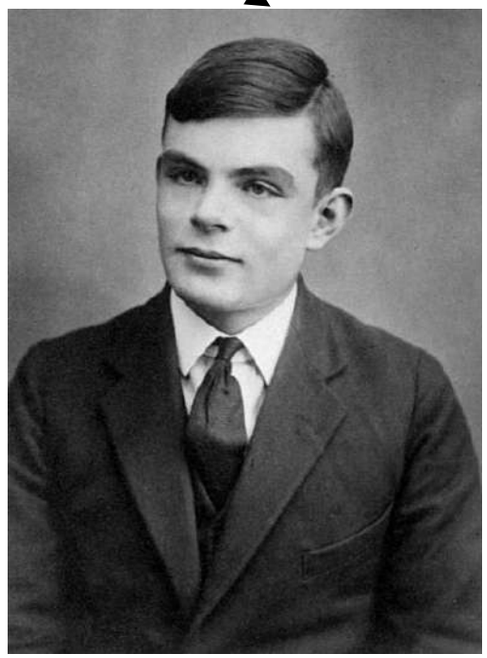
David Hilbert
1862-1943

Matematiksel Mantık
1921



Kurt Gödel
1906 - 1978

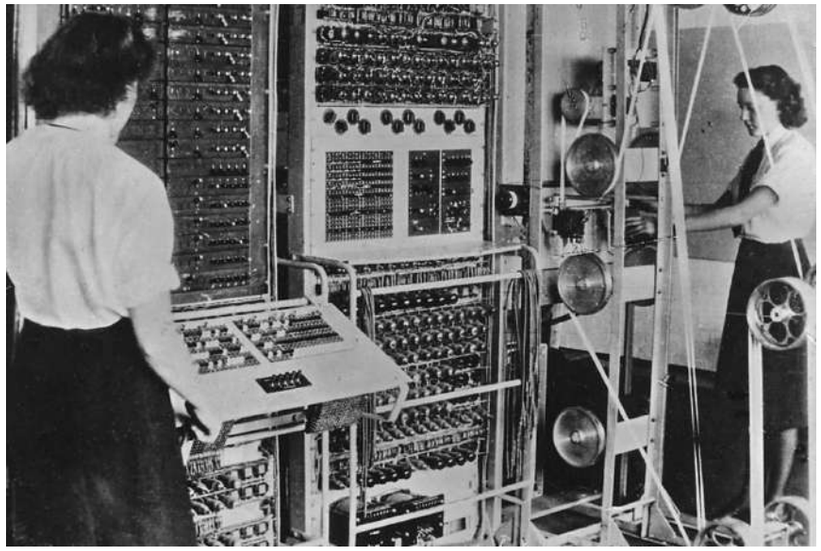
Kapsayamamazlık
(Incompleteness)
1931



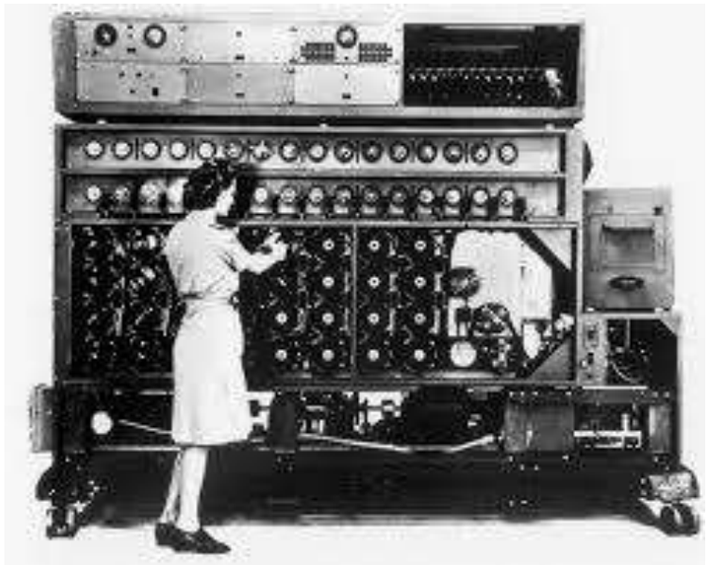
Alan Turing
1912-1954

Kararverilemezlik
(Undecidability)
1936

Somutlaşan Hesaplanabilirlik 1940 - 1950



COLOSSUS



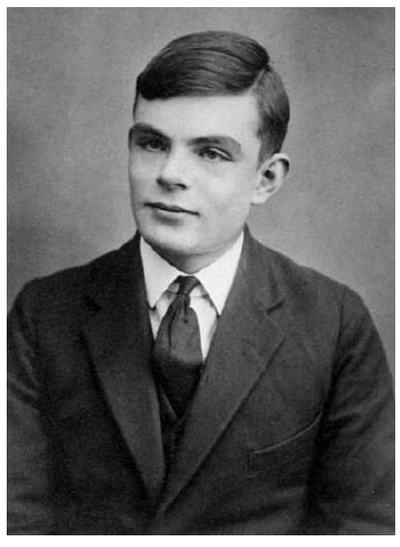
BOMBE

İLK PROGRAMLANABİLİR BİLGİSAYARLAR 1942 - 1945
BLETCHLEY PARK - LONDRA, İNGİLTERE

Alanzo Church
1903 - 1995

$\lambda - calculus$

1936



Alan Turing
1912-1954

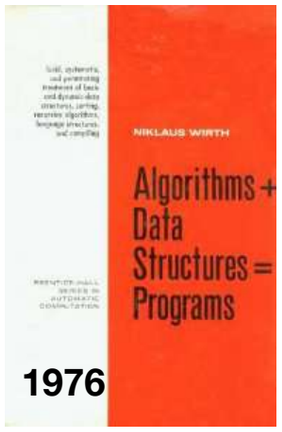
Turing Machine

1936

Church-Turing Tezi:
Doğal sayılar üzerinde hesaplanabilir her fonksiyon Turing makinasında bir metod ile ifade edilir.

Enigma
II. Dünya Savaşı - Alman Şifreleme Cihazı

Programlama Dilleri ve Yazılım Endüstrisi …



FORTRAN'1954; Lisp'1958; ALGOL'60; PASCAL ; C'73; PASCAL'7?;
C++ '82; JAVA'91; PYTHON'00 …….

YAZILIM = ALGORİTMALAR + VERİ YAPILARI

Harizmi
(Al-Khwarizmi)
780 - 850

Algoritma:
Bir problemin çözümü için uygulanacak sıralı adımlardan oluşan yöntem.

Veri:
Bir algoritmanın kullandığı verilerin gösterim şekli

Bir algoritmanın ne kadar iyi olduğunu nasıl ölçeriz?

Hesaplamanın Karmaşıklığı (Computational Complexity)

ON THE COMPUTATIONAL
COMPLEXITY OF ALGORITHMS

BY

J. HARTMANIS AND R. E. STEARNS

I. **Introduction.** In his celebrated paper [1], A. M. Turing investigated the computability of sequences (functions) by mechanical procedures and showed that the set of sequences can be partitioned into computable and noncomputable sequences. One finds, however, that some computable sequences are very easy to compute whereas other computable sequences seem to have an inherent complexity that makes them difficult to compute. In this paper, we investigate a scheme of classifying sequences according to how hard they are to compute. This scheme puts a rich structure on the computable sequences and a variety of theorems are established. Furthermore, this scheme can be generalized to classify numbers, functions, or recognition problems according to their computational complexity.

Bir hesaplamanın zorluğunu nasıl ifade edebiliriz?

Zaman (time) ve yer (space) karmaşıklığı kavramları Hartmanis ve Stearn'1965

Verimli Algoritma (Efficient algorithm):

Edmunds'1965: Bir algoritma, girdisi olan veri büyüklüğünün polinomu zamanında çıktısını üretebiliyor ise verimlidir ! (efficient algorithm)

Zaman ve yer karmaşıklığını teorik olarak ifade etmek için matematiksel bir gösterim gerekli !

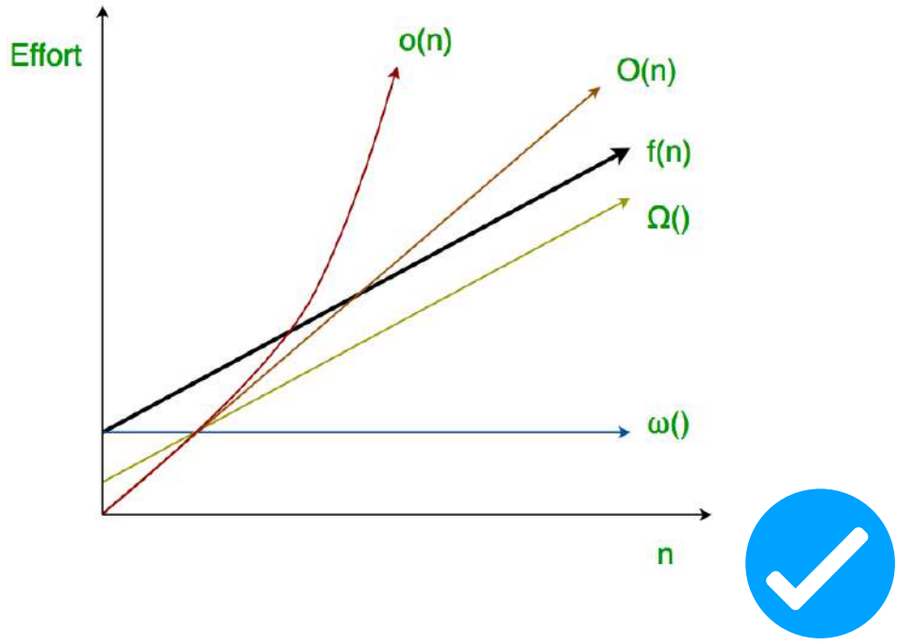
Asimptotik Notasyon

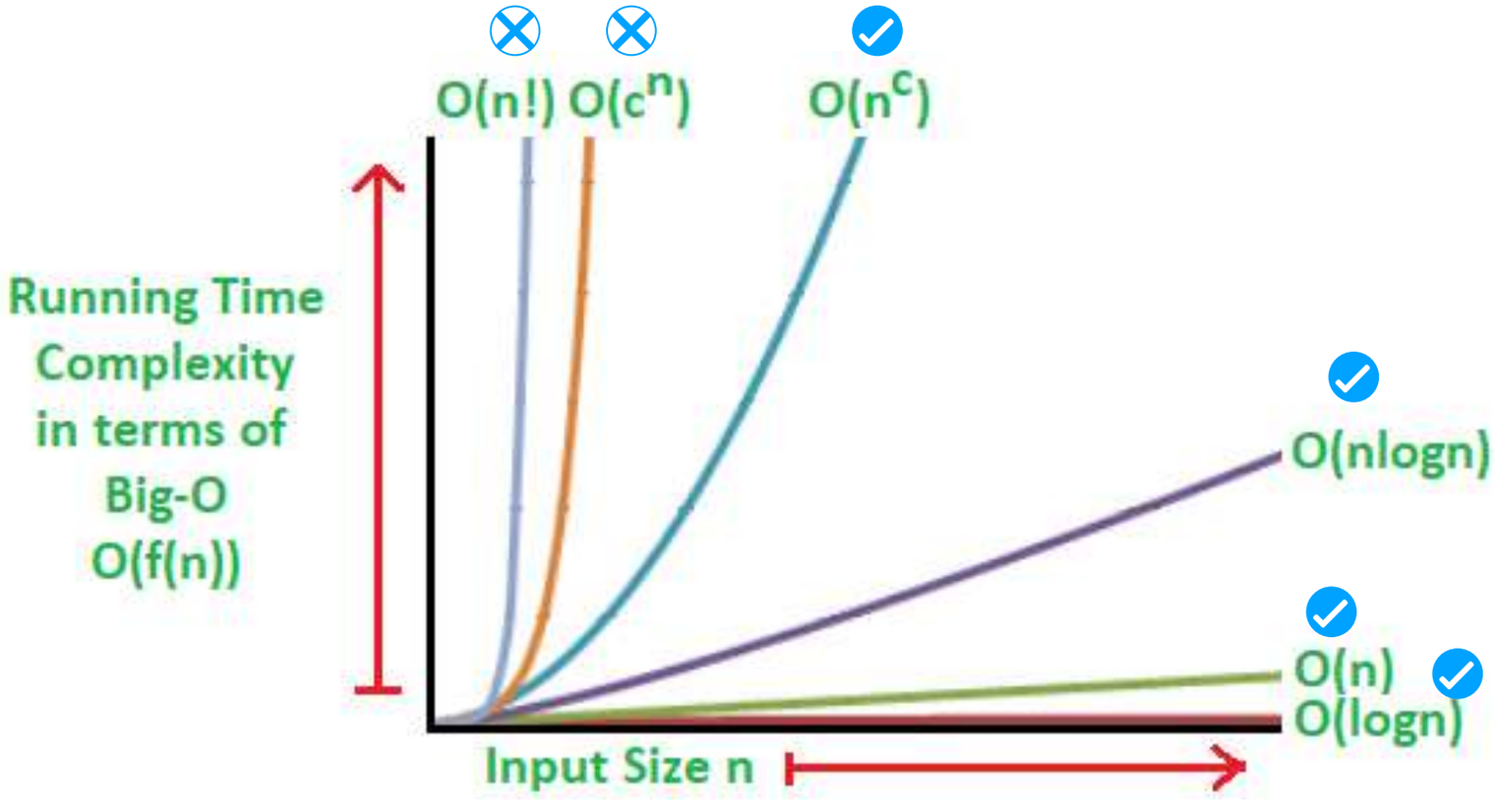
Veri miktarı büyüdükçe algoritmanın çalıştıracağı adımların sayısı ve kullanması gereken ekstra hafıza gereksinimi nasıl büyüyor?

Asimptotik Notasyon

Bachman (1894), Landau (1909):
Hardy&Littlewood(1914); Knuth (1970s)

$$O(), o(), \Omega(), w(), \Theta()$$





Teorik olarak bir problemi, veri miktarının polinomu ile ifade edilen sayıda adım ile çözebilen bir algoritma var ise, o problem verimli olarak çözülebilir (efficiently computable) kabul edilir..

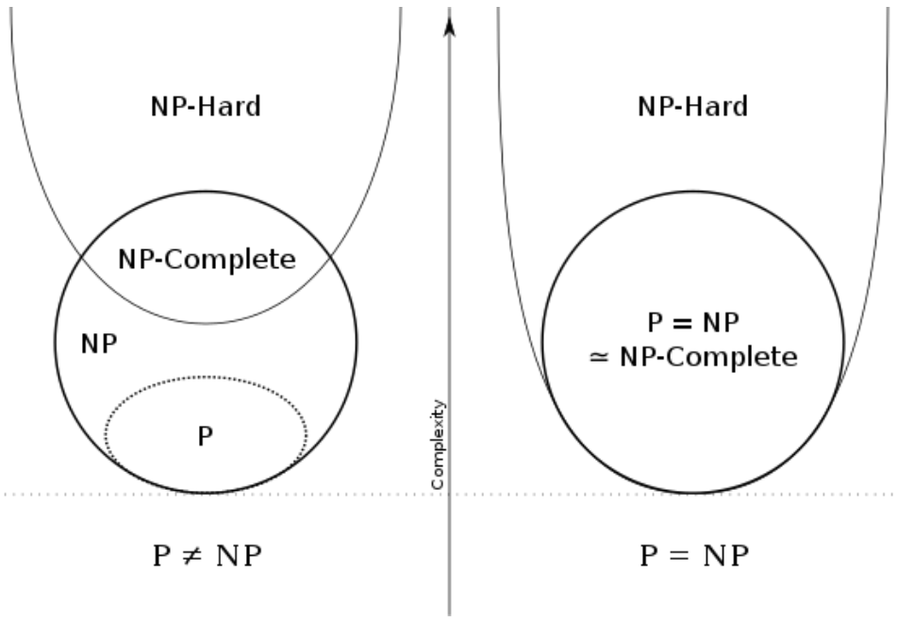
Bazı problemleri çözmek çok zor !

Zor problemler (NP-complete, NP-hard):
Polinom zamanda çalışan bir çözümü bulunamamış problemler.

Stephan Cook
1971 (3-SAT)

Leonid Levin
1971

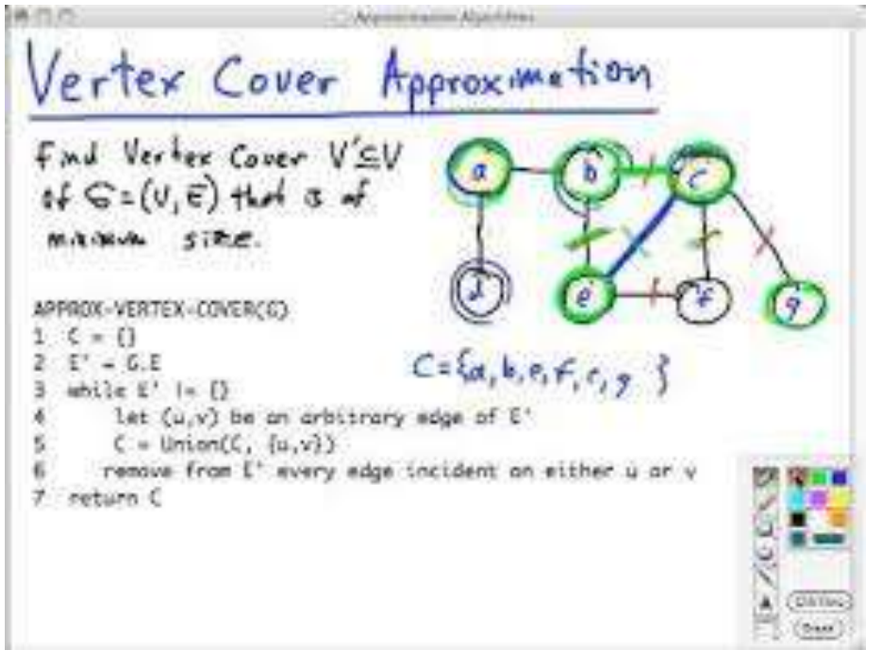
Richard Karp
1972
3-SAT ile indirgenebilir
21 başka problem

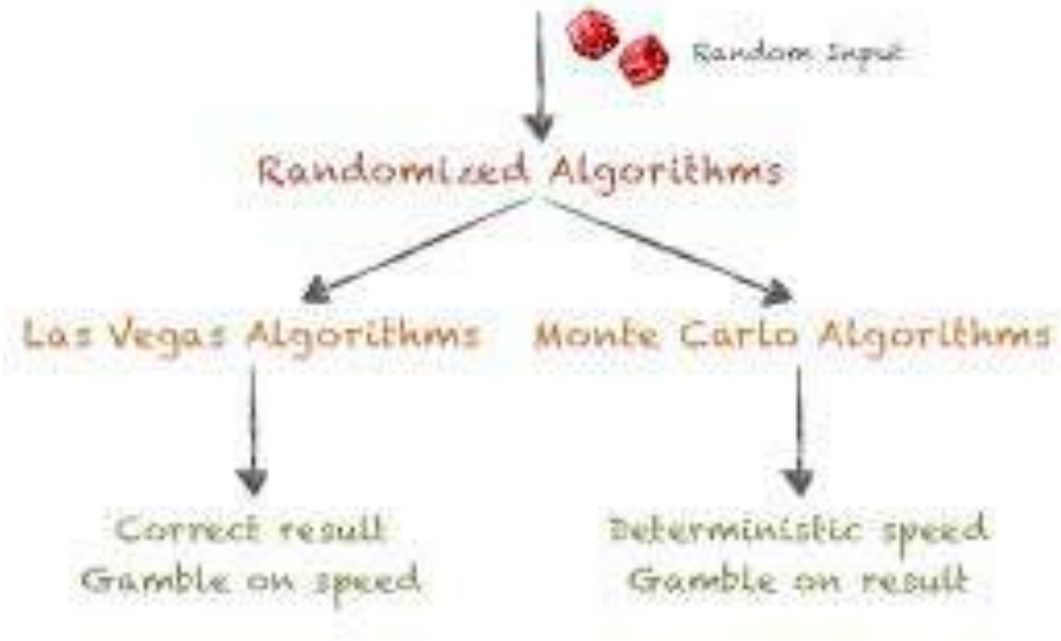


İlginç bir şekilde bu problemlerden (NP-complete) herhangi birisi için çözüm üretilirse hepsi çözülmüş olur (reducability).

Maalesef günlük hayatta en çok ihtiyaç duyduğumuz birçok problem bu sınıftadır !

Zor olsa da birşeyler yapmak lazım elbette …





Yaklaşık algoritmalar
approximation algorithms

Rasgelelik tabanlı algoritmalar
randomized algorithms

Bazı zorluklar da teknik kısıtlardan kaynaklı

*** Akan veri algoritmaları (streaming algorithms)**
*** Dış-hafıza algoritmaları (external memory algorithms)**

Peki, hesaplama kuramına hakimiyet programcıya ne kazandırır ?

Kuramsal temellere hakim bir programcı …

Olası alternatif çözümleri kod yazmadan karşılaştırabilir…

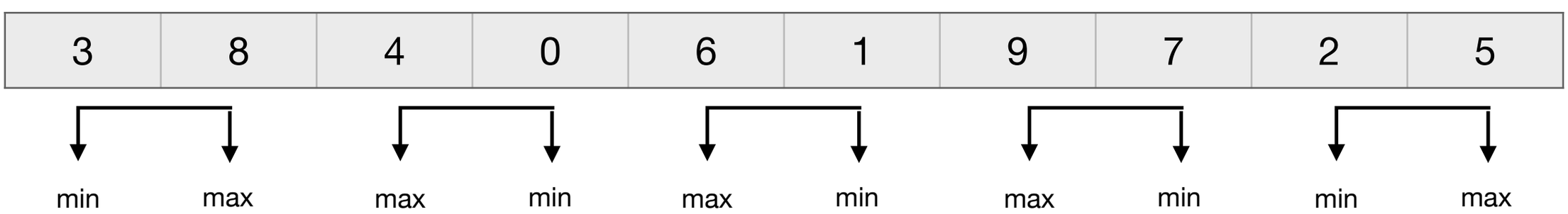
SORU: k adet sayı içerisinde en küçüğünü nasıl buluruz?

3	8	4	0	6	1	9	7	2	5

CEVAP 1 : Sayıları bir sıralama algoritması ile dizerek

CEVAP 2 : Sırayla tüm sayıları kontrol ederek

SORU: k adet sayı içerisinden hem en küçüğünü hem en büyüğünü nasıl buluruz? (2k yerine 1.5k karşılaştırma)



min

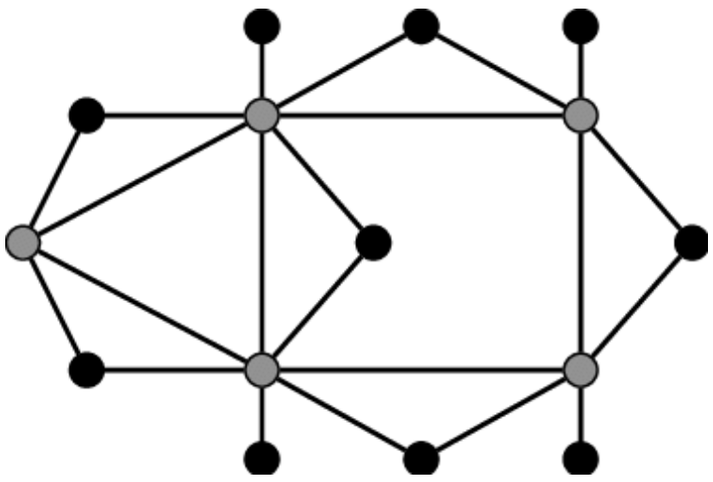
3	0	1	7	2

max

8	4	6	9	5

Kuramsal temellere hakim bir programcı …

Bir problemin zorluğunu farkeder, olmayacak çözümler için beyhude uğraşmaz, seçenekler arasından en iyisini bulmayı bilir…

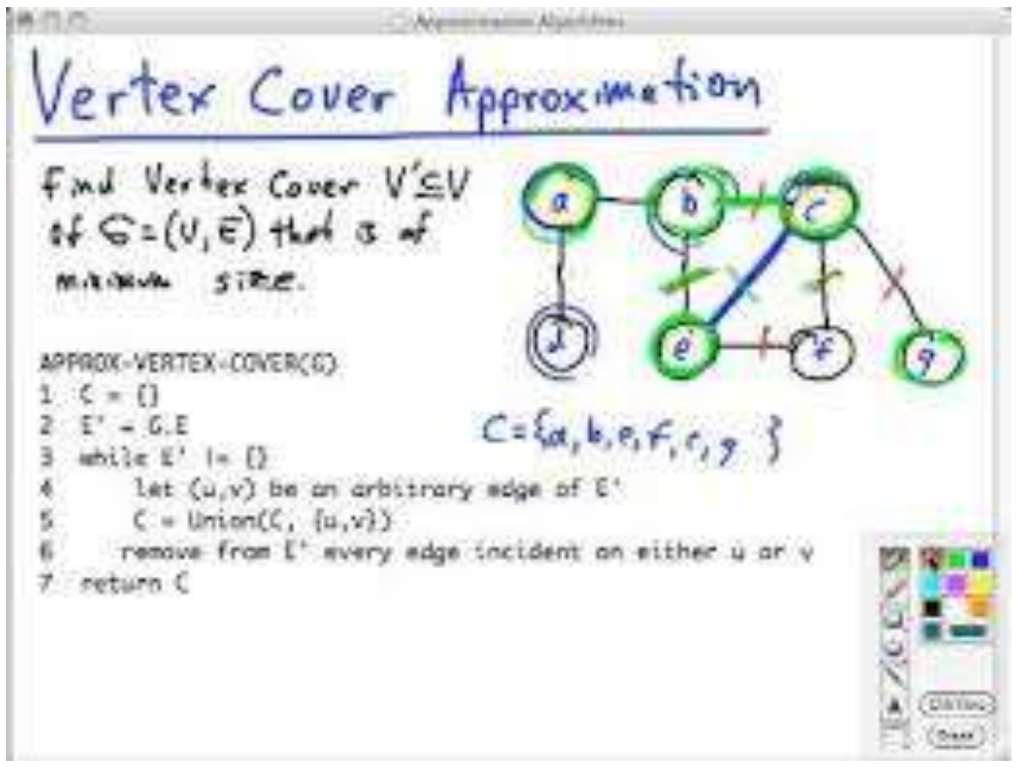


Şehirdeki tüm yolları görecek şekilde kavşaklara kamera yerleştirilecek. Bir kavşaktaki kamera o kavşağa bağlı tüm yolları gösteriyor.
Verilen bir haritaya göre minimum kamera sayısı ?

ZOR bir problem (Vertex-cover, NP-complete)

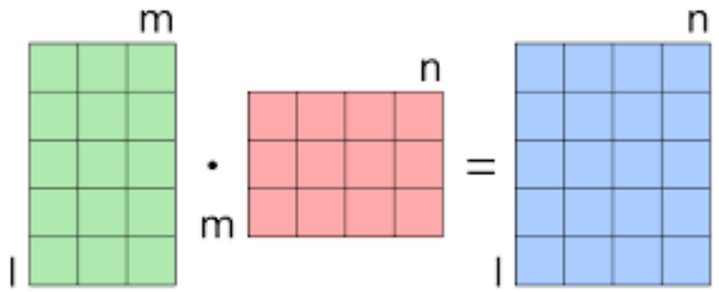
Minimumu bulmak mümkün değil, en azından bugüne kadar hiç olmadı :)

Ama yaklaşık çözümler var, ve bunlar minimumdan en çok ne kadar kötü olduğunu garanti ediyor.



Kuramsal temellere hakim bir programcı …

Programın çalışma süresini azaltmak için rastgelelik tabanlı bir çözümü değerlendirebilir …



Çok büyük iki matrisin çarpımı için özel bir donanım kullanılıyor.
Bu donanımın ürettiği sonucun doğruluğunu en ucuz nasıl test edeceğiz ?

Freivald's Randomized Verification

$$\begin{aligned}\vec{P} &= A \times (B\vec{r}) - C\vec{r} \\ &= (A \times B)\vec{r} - C\vec{r} \\ &= (A \times B - C)\vec{r} \\ &= \vec{0}\end{aligned}$$

$$\Pr[\vec{P} \neq 0] = 0$$

Kuramsal temellere hakim bir programcı …

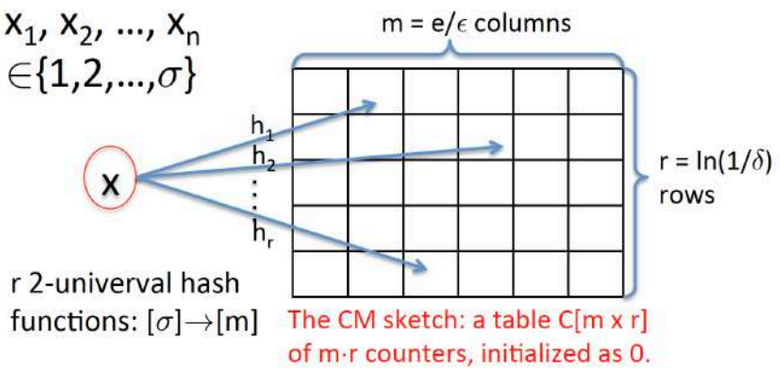
Mevcut kaynaklara en uygun çözümleri üretebilir …

Hızla akan bir veri üzerinde belli bir yüzdeden daha çok geçenleri tesbit etme…

Tüm veriyi kaydetme imkanı yok, eldeki hafıza sonsuz değil ?

The Count-Min Sketch

- Input stream: $x_1, x_2, \ldots, x_n$
- Each element $\in\{1,2,\ldots,\sigma\}$

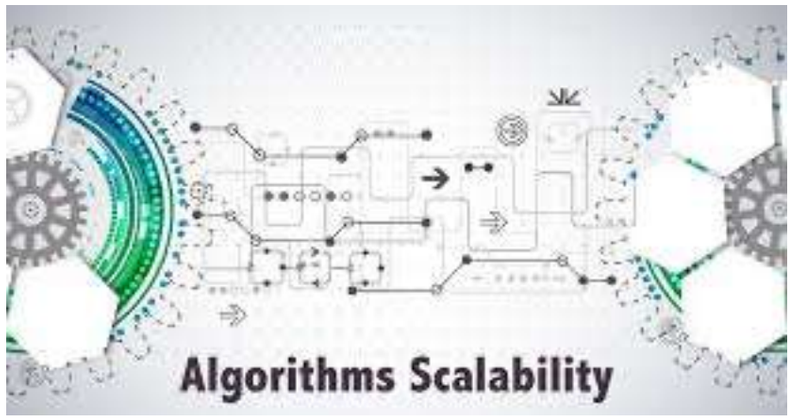


Upon receiving an element x:

For i = 1,2,…,r

C[i, h_i(x)] ++;

Kuramsal temellere hakim bir programcı …



Veri miktarı arttıkça programının nasıl etkileneceğini bilir. Buna göre planlama yapabilir. ÖLÇEKLENEBİLİR çözümler üretir…

Işin teorisine hakimiyet çok iyi bir yazılımlar geliştirmemizi sağlar mı ?

HAYIR!
Özellikle büyük veri çağında ve hesaplama platformlarının bu denli özelleştiği günümüzde, daha da hayır !

Teori gerekli, ama yeterli değil …

Teoride, teori ile pratik aynıdır, ama pratikte değildir :)

In Theory There Is No Difference Between Theory and Practice, While In Practice There Is

- **Teorik kabuller, pratikle uyuşmayabilir !**
  (her atomik işlem aynı süre almaz, değişken word-size mümkün değil, vb…)

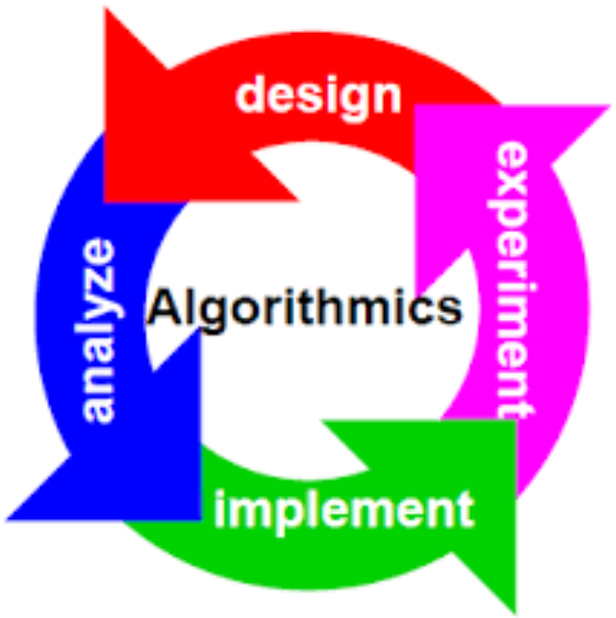
- **Karmaşıklık analizi en kötü senaryoya göre yapılır, gerçek hayatta her şey o kadar da kötü olmayabilir.**

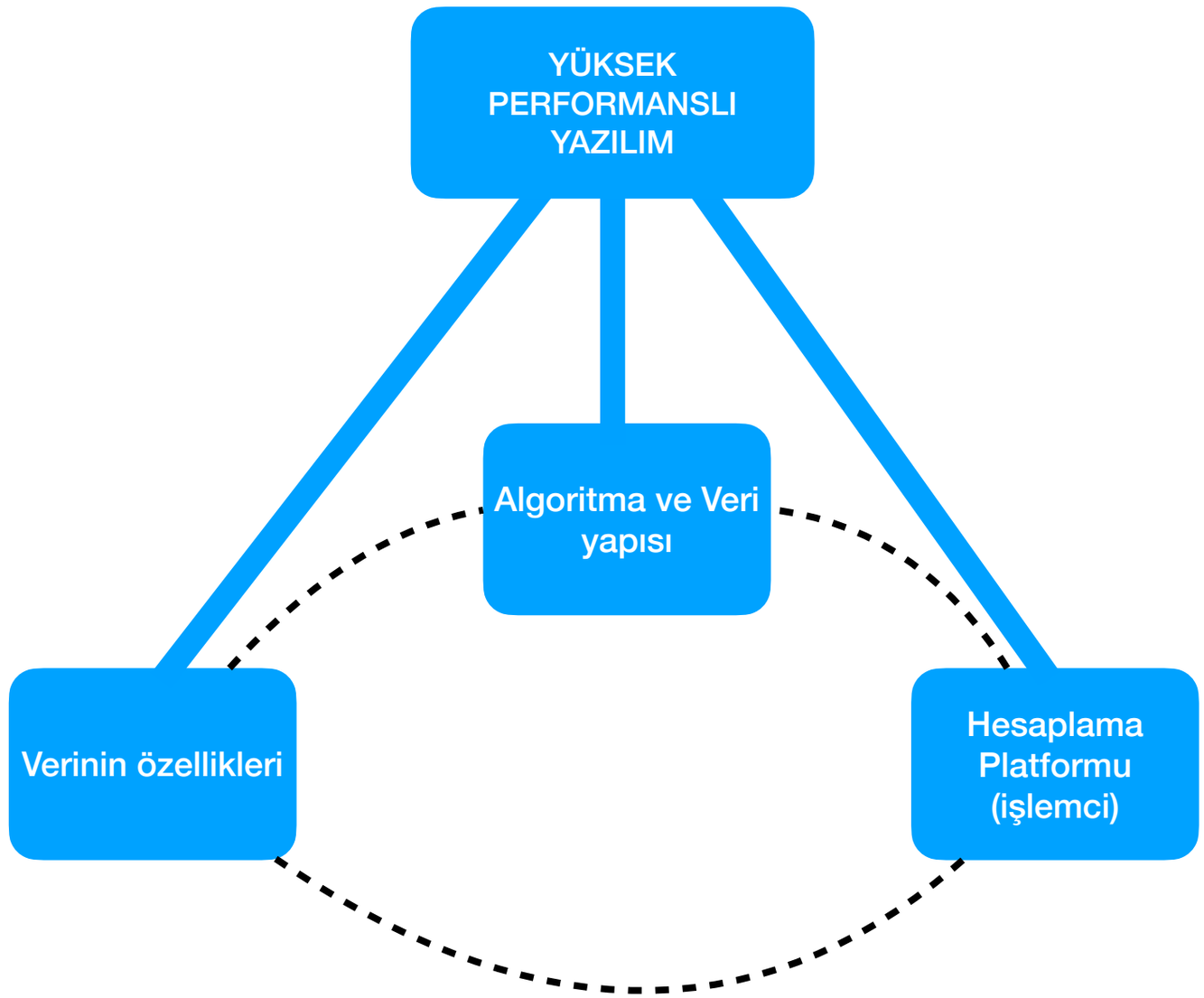
- **Asimptotik notasyondaki gizli sabitler, pratik uygulamada çok önemli olabilir.**

- **Algoritmanın üzerinde çalışacağı işlemcinin (platformun) özellikleri fark yaratır**
  (pipelining, latency, data locality, branch prediction, instruction cycles)

- **Üzerinde çalışılacak verinin özellikleri, pratikteki senaryo ile teorik senaryonun farklılıkları…**

Algoritma Mühendisliği





Algoritma Mühendisliği - kısa tarihçe…

70’lerde teori ağırlıkta

80’lerde bilgisayarın yaygınlaşması, teorinin pratikte kullanımının artışı

80’lerin sonunda teorik sonuçlar ile pratik gerçekler arasında makasın açılması

90’ların sonu, 2000’lerin başı algoritma mühendisliğinin ortaya çıkışı

Workshop on Algorithm Engineering

Venice, Italy

September 11-13, 1997

ESA- Track B,
experimental algorithms

ALENEX 99

Workshop on Algorithm Engineering and Experimentation

January 15-16, 1999
Omni Hotel, Baltimore, Maryland

2000’den beri gelişen işlemci ve hesaplama paradigmasının pratik sonuçları

Bir örnek ile ....

Dizgi eşleştirme problemi (pattern/string matching)

Text : A A B A A C A A D A A B A A B A

Pattern : A A B A

A A B A

A A B A

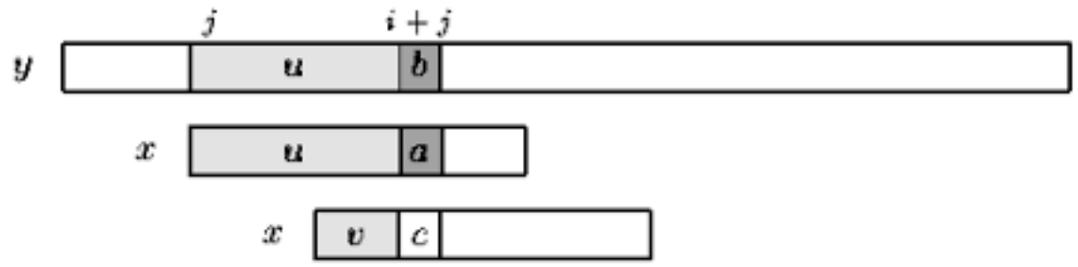
A A B A A C A A D A A B A A B A
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

A A B A

Pattern Found at 0, 9 and 12

Temel çözüm : $O(n^2)$

Optimal çözüm : $O(n)$



1977- Knuth-Morris-Pratt

…. birçok varyant

Ricardo Baeza-Yates and Gaston H. Gonnet

A New Approach to

Text Searching

String searching is a very important component of many problems, including text editing, bibliographic retrieval, and symbol manipulation. Recent surveys of string searching can be found in [4, 18].

The string-matching problem consists of finding all occurrences of a pattern of length m in a text of length n. We generalize the problem allowing *don't care* symbols, the complement of a symbol, and any finite class of symbols. We solve this problem for one or more patterns, with or without mismatches. For small patterns the worst-case time is linear in the size of the text (we say that a pattern is small if m is bounded by a constant).

GÖZLEM :

İşlemcilerde bit manipülasyon işlemleri çok hızlı çalışır.

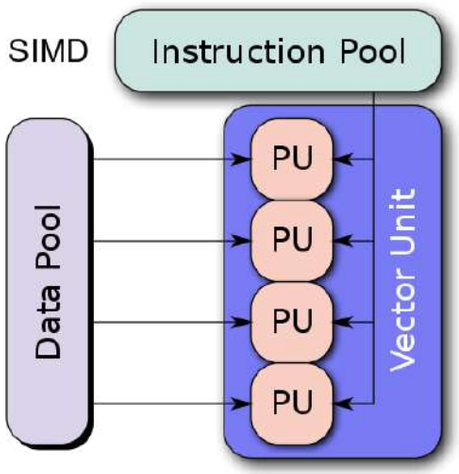
1992- Baeza-Yates & Gonnet: Shift-Or algoritması;

… takip eden yıllarda BNDM, SBNDM, ve diğer varyantlar

- **Diğer lineer zamanlı çözümlerden çok daha hızlı**
- **Fakat aranan dizgi bilgisayarın yazmaçından (register) uzun olmamalı ?**

2008 - Külekci - BLIM algoritması: Uzunluk kısıtına çözüm

Bir örnek ile ….

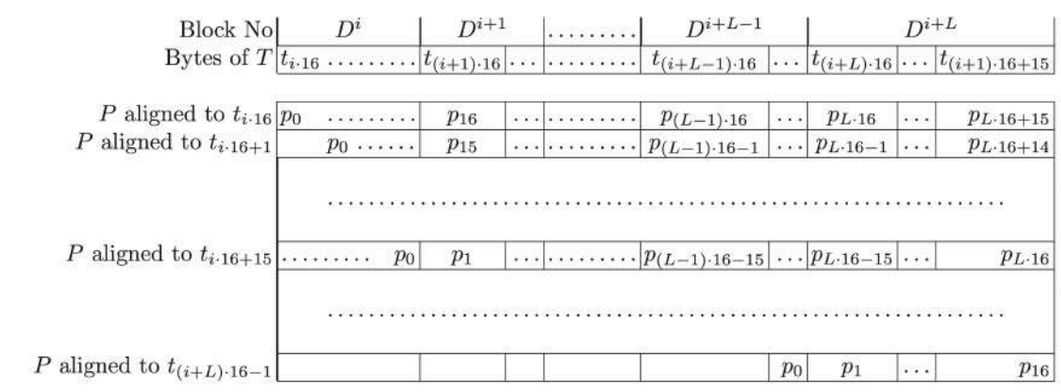


GÖZLEM: Aslında işlemcilerde çok daha uzun yazmaçlar var
- **SIMD teknolojisi MMX, SSE, AVX, …**
- **Dizgi eşleştirme için kullanılamaz mı ?**

2009 - Külekci - SSEF algoritması,
Prague Stringology

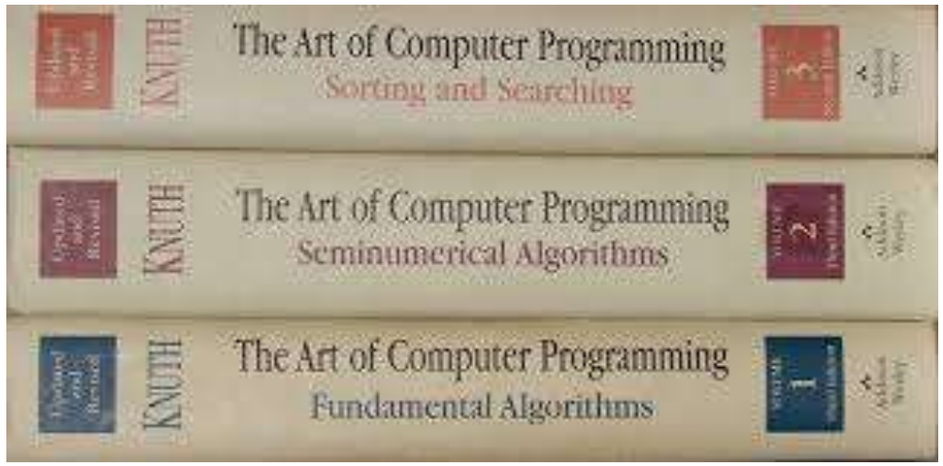
2011- EPSM - Faro & Külekci - ALENEX;
2018-EPSM2 - Aydoğmuş &Külekci - VECPAR

PACKED STRING MATCHING



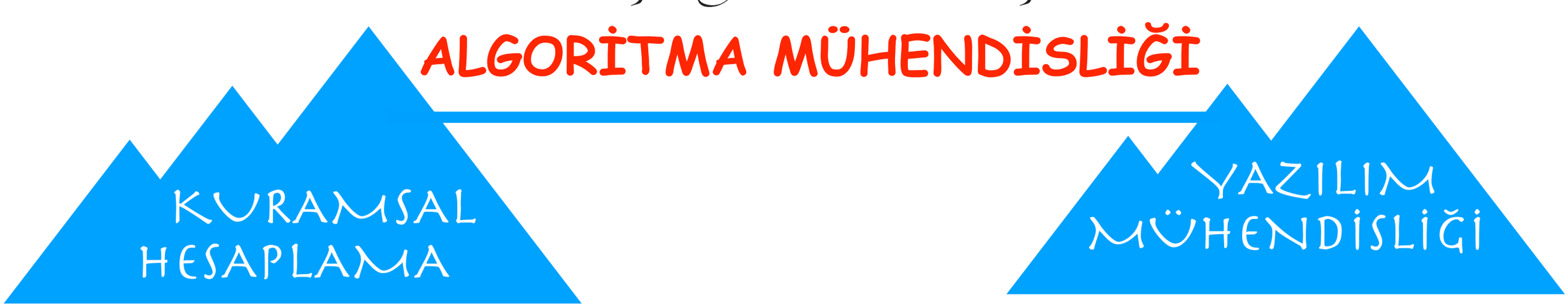
SSE içindeki gömülü dizgi eşleştirme komutundan (instruction) çok daha hızlı ?
- latency, throughput …

SONUÇ:
İşlemcilerdeki teknolojik gelişmeleri dikkate alarak geliştirilen çözümler çok daha verimli olabilir.



Programlama sanatı ...

Teori ile pratiğin arasındaki köprü ....



TEŞEKKÜRLER .....