

Veri Modeli Nedir?

- Veri modelleri, aynı kümeye ait sayılabilecek veriler arasında ilişkisel ve sırasal düzeni gösteren kavramsal yaklaşım yöntemleridir.
- Uygun veri modeli seçilmesi ve kullanılması “uygulama için en etkin programın yazılmasını sağlar” denilebilir.
 - Etkinlik:
 - bazen hız
 - bazen de gereksinimi duyulacak bellek alanı miktarıdır.

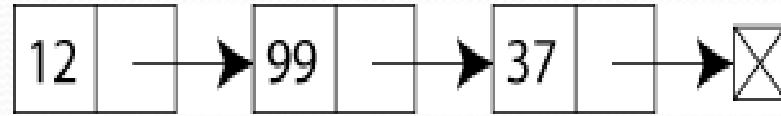
Çeşitli Veri Modelleri

- Bağlantılı Liste (Linked List)
 - Ağaç (Tree)
 - Graf (Graph)
 - Durum Makinası (State Machine)
 - Veritabanı-İlişkisel (Database Relational)
 - Ağ Bağlantı (Network Connection)
- Hazır veri modellerini karşılaştırma:
- Dizi
 - ArrayList, List
 - Hashtable, Dictionary

Hangi Veri Modeli

- Uygulamanın Türü
 - İstemci tabanlı mı? Sunucu/İstemci tabanlı mı?
- Uygulamanın nerede çalışacağı
 - Web mi, Bilgisayar mı, Gömülü sistem mi?
- Hangi Veri Türü üzerine İşlem?
 - Metin, Ses, Video
- Ne kadar Bellek Gerekli?
 - RAM
- Hız Önemli mi?
 - CPU
- Çoğu zaman, programın çalışma hızıyla bellek gereksinimi miktarı doğru orantılıdır denilebilir.

Bağlantılı Liste



- Bağlı liste herhangi bir tipten node'ların (düğümelerin) yine kendi tiplerinden düğümlere işaret etmesi (point) ile oluşan zincire verilen isimdir.
- Buna göre her düğümde kendi tipinden bir pointer olacak ve bu pointerlar ile düğümler birbirine yukarıdaki şekilde bağlanacaktır.

Bağlantılı Liste

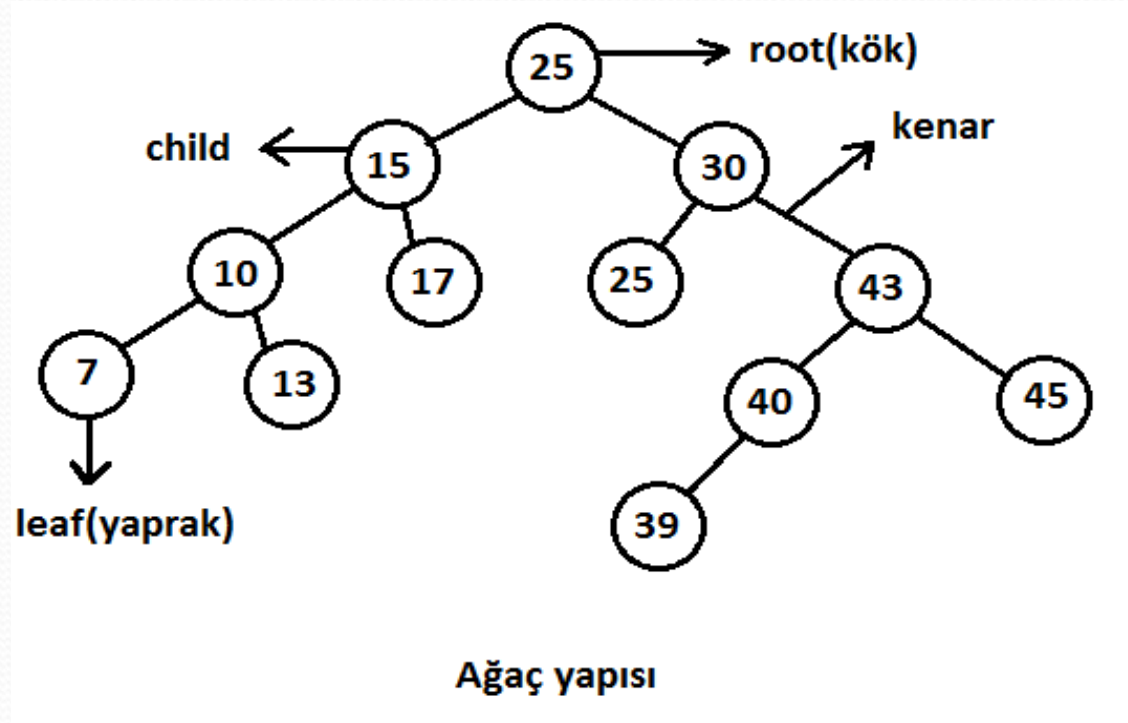
- Avantajları

- Veri büyüklüğü önemli değildir. Bir tahsis işlemi gerektirmez.
- Silme, düzeltme, ekleme işlemleri için tekrar bellek düzenlemesi gerektirmez.
- Dizilere göre belleği daha etkin bir şekilde kullanır.

- Dezavantajları

- İşaretçileri tutmak için ek alan
- Yapı sırasaldır. Arama bakımından dizilerden pek farkı yoktur. Hatta daha yavaş bile olabilir.

Ağaç

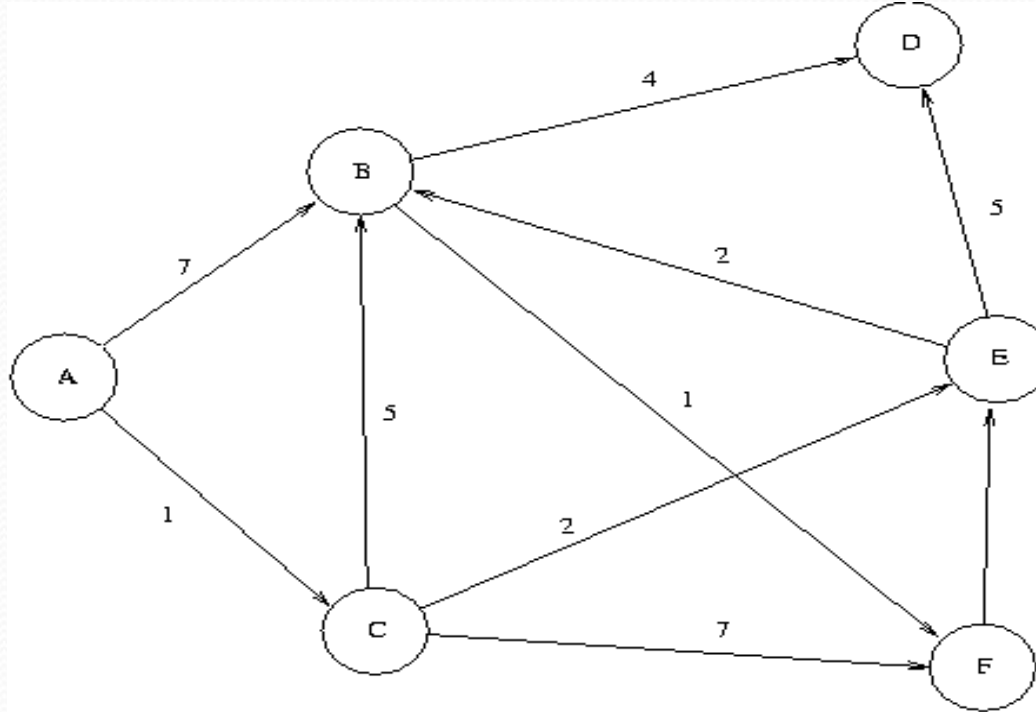


- Verilerin birbirine sanki bir ağaç yapısı oluşturuyormuş gibi sanal olarak bağlanmasıyla elde edilen hiyerarşik yapıya sahip veri modelidir.

Ağaç mı? Dizi mi? Bağlı Liste mi?

- Arama süresi
 - Dizi ve Bağlı Listeye göre çok hızlıdır.
- Ekleme
 - Dizi ve Bağlı Listeye göre yavaşlar, çünkü her ekleme işleminde arama işlemini yapmak gerekir.

Graf

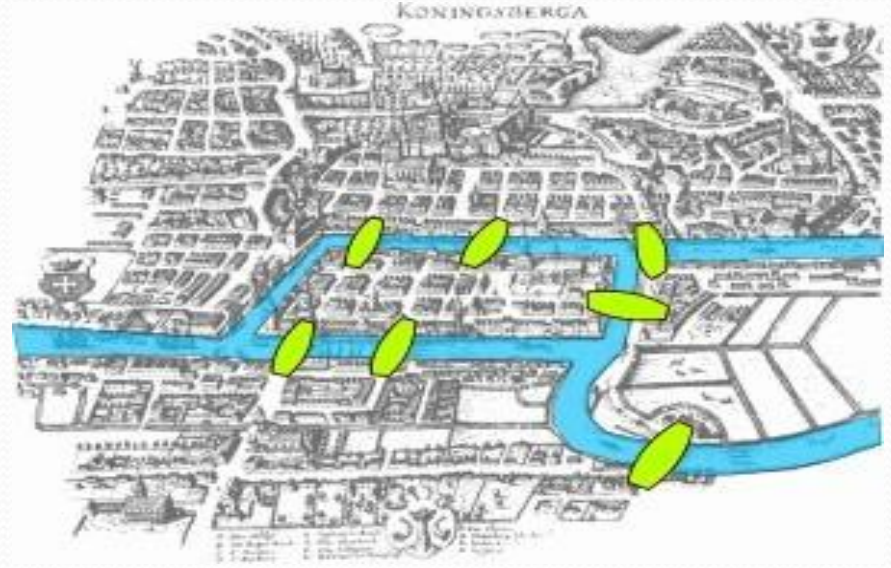


- Şehrin trafik ve su akışının sağlanması,
- Taşıma şirketlerinin en verimli taşıma şekli
- Network bağlantılarında yüksek başarımlar elde edilmesi

İlk Graf!!!

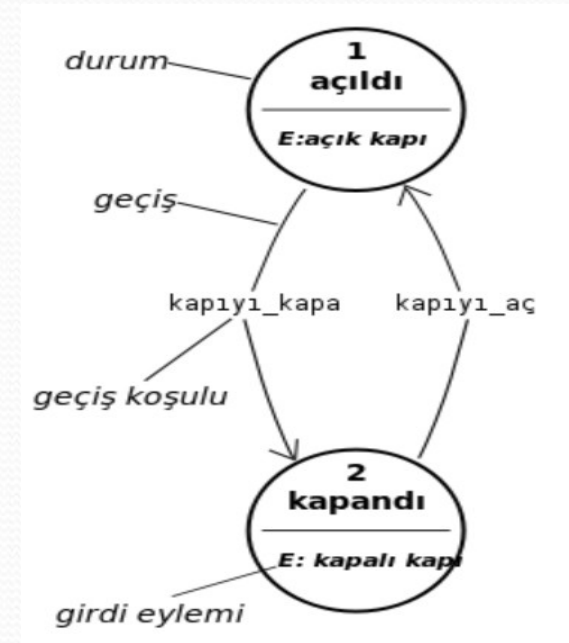
- Yedi Königsberg Köprüleri ve 1736 yılında yayımlanarak Leonhard Euler grafik teorisi tarihinin temeli olarak kabul edilir.

Bütün köprülerden bir ve yalnız bir kez geçmek koşulu ile bir yürüyüş yapılabilir mi?

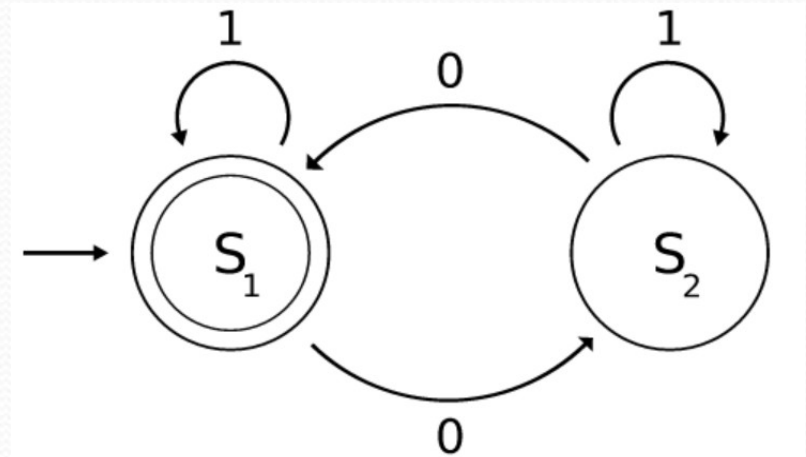


Durum Makineleri

- Sınırlı sayıda durumdan, durumlar arası geçişlerden ve eylemlerin birleşmesiyle oluşan davranışların bir modelidir.



	0	1
s_1	s_2	s_1
s_2	s_1	s_2



Durum Makineleri

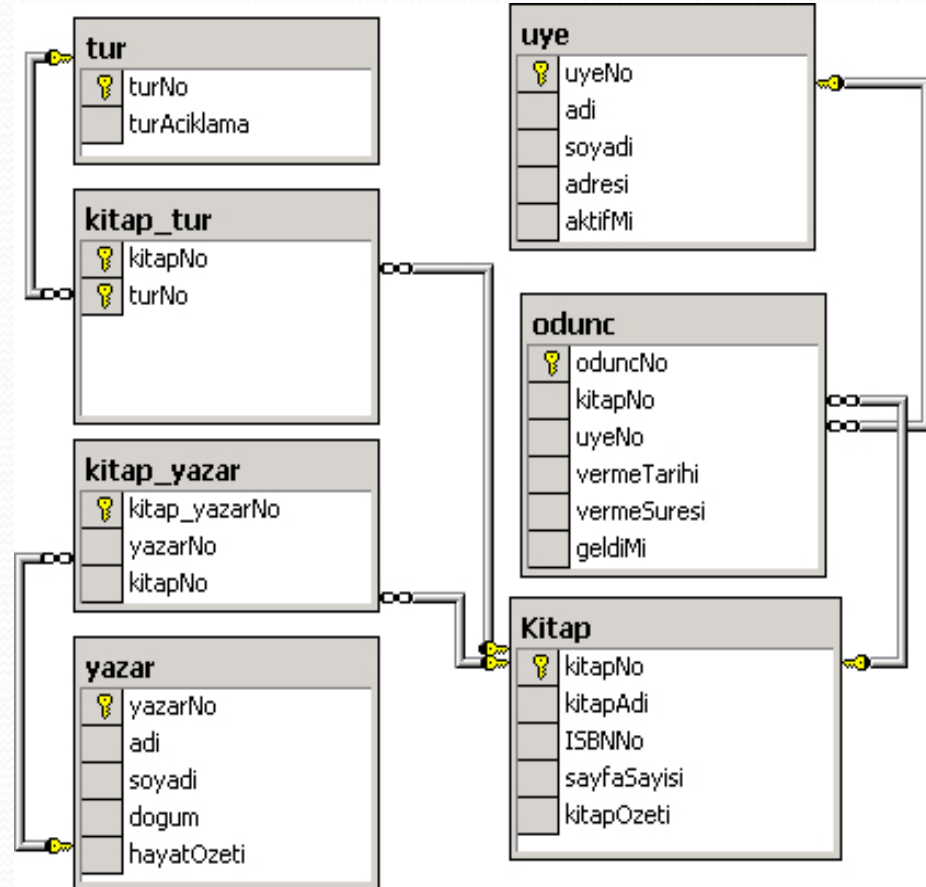
- İşletim sistemlerinde, derleyici/yorumlayıcılarda, kontrol amaçlı yazılımlarda sistemin davranışını durumlara indirger ve durumlar arası geçiş koşullarıyla sistemi ortaya koyar.
- Durum makinesi, yazılım uygulamasında birçok alanda kullanılabilir. Örneğin
 - bir robot kolunun hareketi,
 - şifre çözme,
 - gerçek zamanlı işletim sistemlerinde proses kontrolü
 - genel olarak kontrol alt sistemlerinin yazılımla kotarılması durumlarında çözüm olur.

Durum Makineleri

- Durum makinası veri modeli şeklen yönlü graflara benzer, ancak, birleşme noktaları graflarda olduğu gibi düğüm olarak değil de durum, ayrıtlar da geçiş eğrileri olarak adlandırılır.
- Durumlar arasındaki geçişler, sistemin o ana kadar ki durumlarına ve giriş parametrelerine bağlıdır.

Veritabanı – İlişkisel Veri Modeli

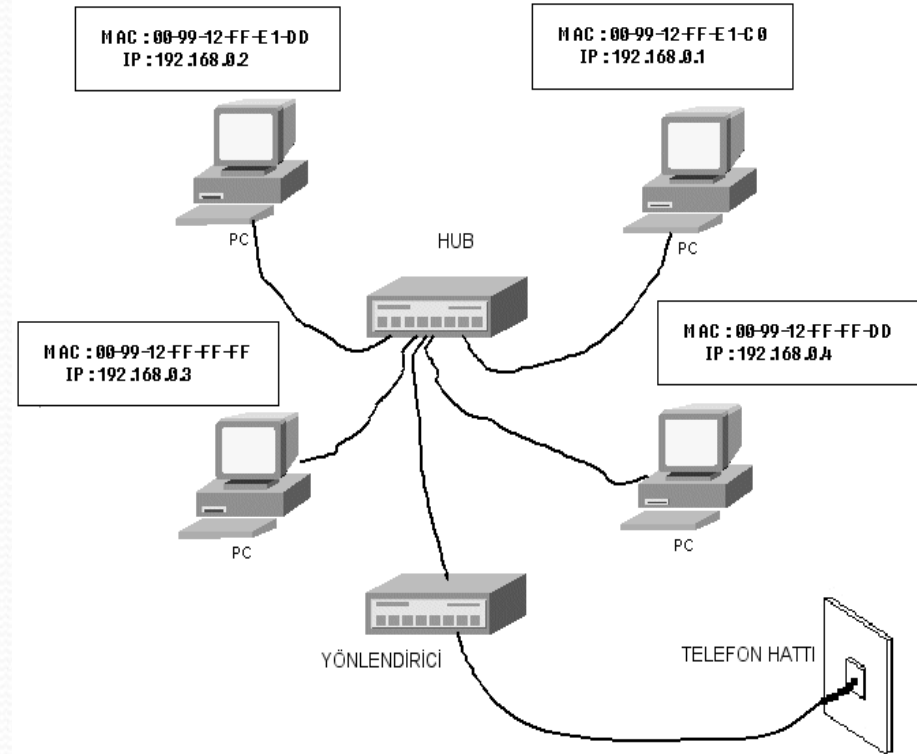
- Veritabanı ilişkisel veri modeli veritabanı uygulamalarında var olan dört beş sınıftan birisidir; veriler şekilde gösterildiği gibi tablolar üzerinden kurulan ilişkilere dayanmaktadır.



Ağ Veri Modeli

- Ağ veri modeli, katmalı ağ mimarilerinde, bilgisayarlar arasında eş katmalar (peer layers) düzeyinde veri alış-verişini sağlayan dilim (segment), paket (packet) ve çerçeve yapılarını ortaya koyar ve iletişim için gerekli davranışı tanımlar.

- IP
- MAC
- İstemci / Sunucu
- HUB - Switch
- Yönlendirici
- ADSL



Dictionary ve Hashtable

- Dictionary Hashtable'ın generic versiyonudur.
- Bu iki sınıfın çalışma prensipleri birbirine çok yakındır. Bir bölgede iki adet veri barındırırlar. Bunlar 'Key' ve 'Value'dur. Verdiğimiz 'Key' değeri unique olmak ve null olmamak zorundadır.
- HashTable ve Dictionary, verilere ulaşırken barındırdıkları unique anahtarları kullanarak ulaşırlar. Böylece aldıkları verilere bir nevi Identity kolonuna bağlamış olurlar. Key değerleri atandıktan sonra asla değişmez.
- Boyut belirtme zorunluluğu yoktur ve arama performansları çok yüksektir.
- IndexOfKey, IndexOfValue, RemoveAt, GetByIndex, SetByIndex gibi metotları barındırarak indekslemeyi daha esnek kılar.

Hash

- Hash, birer anahtar ve o anahtara karşılık gelen verileri tutan yapılardır. Veriler Key -Value (Anahtar – Değer) ilişkisi içinde tutulur.
- Dizilerde arama işlemi için tüm kayıtlara bakılmalı veya direk indis bilinmelidir. Hash fonksiyonunda ise bir adımda istenen kayda ulaşılacak istenmiştir.

Hash Fonksiyonu

Verilen bir anahtar için öncelikle bir indeks oluşturulur.

$\text{index} = f(\text{anahtar}, \text{dizi_boyutu})$

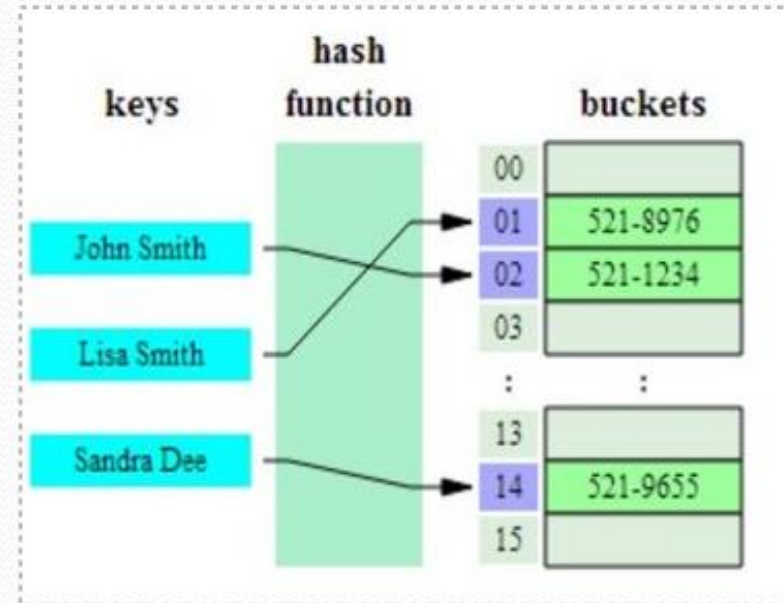
Genellikle bu işlem iki adımda yapılır.

$\text{hash} = \text{hashfunc}(\text{key})$

$\text{index} = \text{hash} \% \text{dizi_boyutu}$

Hashfunc fonksiyonu verilen anahtar için bir sayısal değer üretir.

Mod işlemi hash dizisi içinde kalmamızı sağlar.



Hash Fonksiyonu

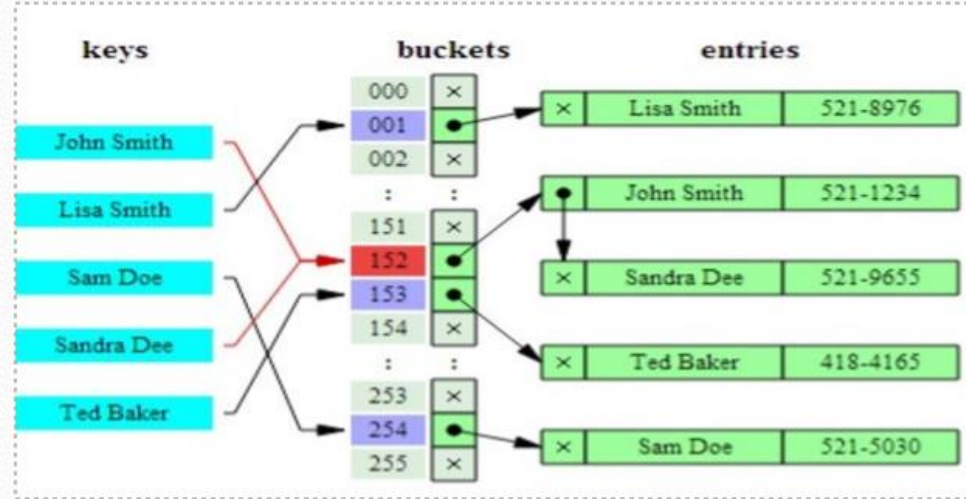
Hash fonksiyonu, değişken uzunluklu veri kümelerini, sabit uzunluklu veri kümelerine haritalayan algoritma veya alt programdır.

Örneğin, bir kişinin ismi değişken uzunlukta ise, tekil tamsayı olarak hash'lenebilir.

Hash fonksiyonundan üretilen bir değer çalışabilir. (Hash Collision)

Hash çakışması, separate chaining (ayrı zincirleme) ile çözülebilir. Bu yöntemde, çakışma olan kayda ekleme yapılır ve bu yapı dizi gibi çalışır.

Farklı çözümleme yöntemleri de vardır.



Kullanım Yerleri

- Associative arrays (İlişkisel diziler):
 - Hash Table
 - .Net ve Java framework'leri tarafından desteklenir.
- Database Indexing (Veritabanı İndeksleme) Caches
- Bloom filters
- Finding duplicate records (Yinelenen kayıtları bulma)
- Protecting data (Veri koruma)
 - Cryptographic hash function (Kriptografik hash fonksiyonu)
 - Finding similar records (Benzer kayıtları bulma)
 - Finding similar substrings (Benzer alt katarları bulma)

İlişkisel dizilerin karşılaştırılması

- Dizi
 - Doğrusal arama
- ArrayList
 - Arama
- Dictionary
 - Hızlı arama sağlar.
- Hash Table
 - Arama hızı optimize eder. Ancak, hız bakımından Dictionary'den yavaştır.



Algoritmik Program Tasarımı

- Program Tasarımı
- Algoritma
 - Kaba Kod
 - Akış şeması
 - N-S (Nassi-Schneiderman) Şeması
 - W-O (Warnier-Orr) Diyagramı
- Algoritma Tarihi
- Algoritma Patent Olur mu?
- Algoritma Örnekleri

Program Tasarımı

.Program tasarımı, verilen bir problemin bilgisayar ortamında çözülecek biçimde adım adım ortaya koyulması ve herhangi bir programlama aracıyla (programlama dili veya tasarım araçlarıyla) kodlanması sürecidir.

.Program tasarımı için

- Mevcut / Eski programın incelenmesi
- Anahtar kelimeler belirlenmesi
- Mevcut open-source ve ticari uygulamaların incelenmesi
- Algoritmanın tasarlanması / Akış şemalarının çizilmesi / Pseudo (Kaba) kodun yazılması
- Uygun programlama dilinin seçimi
- Kodun gerçekleştirilmesi

Algoritma

.Algoritma, matematikte ve bilgisayar biliminde bir işi yapmak için tanımlanan, bir başlangıç durumundan başladığında, açıkça belirlenmiş bir son durumunda sonlanan, sonlu işlemler kümesidir.

.Yani belli bir problemi çözmek veya belirli bir amaca ulaşmak için çizilen yola algoritma denir.

- Tüm programlama dillerinin temeli algoritmaya dayanır.
- Aynı zamanda algoritma tek bir problemi çözecek davranışın, temel işleri yapan komutların veya deyimlerin adım adım ortaya konulmasıdır ve bu adımların sıralamasına dikkat edilmelidir.

Algoritma

Algoritma kelimesi, Özbekistan'ın Harezmi, bugünkü Türkmenistan'ın Khiva kentinde doğmuştur.

Algoritma sözcüğü Fars âlim Ebu Abdullah Muhammed bin Musa el Harezmi'nin Latince isminden kaynaklanır.

Algoritma, programlama dilinden bağımsız yaratılır ve birden fazla programlama diline uygulanabilir.

Ayrıca, On rakamdan oluşan rakamlama sistemi yine, Harezmi tarafından tanıtıldığı için Arap Rakamları olarak adlandırılır ve **0'ı bulan kişi** olarak tanınır. Bu aynı zamanda bilgisayarın çalışma mantığı olan ikilik sisteminde temelidir.



Algoritmanın etkinliği

Bir algoritmanın etkinliği, uygulama öncesi ve sonrasında iki farklı aşamada analiz edilebilir. Bunlar şunlardır –

- **Öncelikli (Priori) Analiz** - Bu bir algoritmanın teorik bir analizidir. Bir algoritmanın verimliliği, diğer tüm faktörlerin, örneğin işlemci hızının sabit olduğu varsayılarak ölçülür. Bunda, algoritmanın verimliliği, karmaşıklığına göre hesaplanır. Bu sadece yaklaşık bir analizdir.
- **Kaba (Posterior) Analiz** - Bu, bir algoritmanın ampirik bir analizidir. Seçilen algoritma, programlama dili kullanılarak gerçekleştirilir. Bu daha sonra hedef bilgisayar makinesinde yürütülür. Bu analizde, çalışma süresileri toplanıp değerlendirilir. (C# için stopwatch sınıfı kullanılabilir.)

Algoritma

Algoritmayı belirtmek için

- Kaba kod (metinsel olarak düz ifade)
- Akış diyagramı
- N-S (Nassi-Schneiderman) Şeması
- W-O (Warnier-Orr) Diyagramı

olmak üzere 4 yöntem kullanılır.

Kaba kod (Pseudo Code)

- .Pseudo kod (Kaba kod), bir programlama dilinden ziyade programcının dilinde yazılmış ön hazırlık kodudur.
- .Gerçek bir programlama dilinin sözdizimi kurallarını taşımaz.
- .Sadece genel olarak algoritmanın işleyişini gösterir.
- .Kullanıcının girdiği dört sayının ortalamasını görüntüleyen algoritma:
 - A0 --> Başla
 - A1 --> Sayaç=0 (Sayaç'ın ilk sayısı 0 olarak başlar.)
 - A2 --> Sayı=? : $T=T+Sayı$ (Sayıyı giriniz. T'ye sayıyı ekle ve T'yi göster.)
 - A3 --> $Sayaç=Sayaç+1$ (Sayaç'a 1 ekle ve sayacı göster.)
 - A4 --> $Sayaç < 4$ ise A2'ye git. (Eğer sayaç 4'ten küçükse Adım 2'ye git.)
 - A5 --> $O=T/4$ (Ortalama için T değerini 4'e böl)
 - A6 --> O'yu göster. (Ortalamayı göster.)
 - A7 --> Dur

Kaba kod

en küçük = Dizinin ilk elemanı

for (ikinci elemandan dizi sonuna kadar)

if (sıradaki eleman < en küçük)

en küçük = sıradaki eleman

en küçük elemanı yazdır veya
gönder

```
1  bulenkucuk(int a[])
2  {
3      int enkucuk = a[0];
4      for (int k=1; k<a.Length; k++)
5          if (a[k]<enkucuk)
6              enkucuk=a[k];
7      return enkucuk;
8  }
```

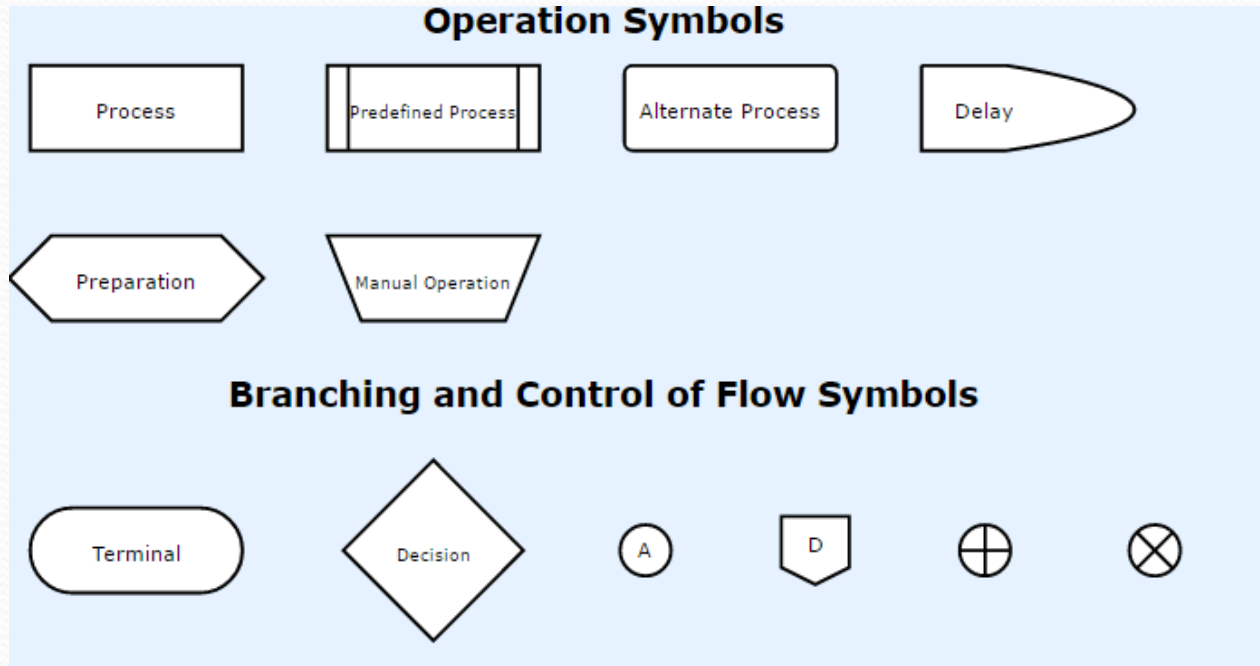
Akış Şeması (Flow Chart)

.Akış şeması algoritmaları ve işlemleri birbirine oklarla bağlı değişik tiplerdeki kutular içerisinde gösteren yaygın bir şema tipidir.

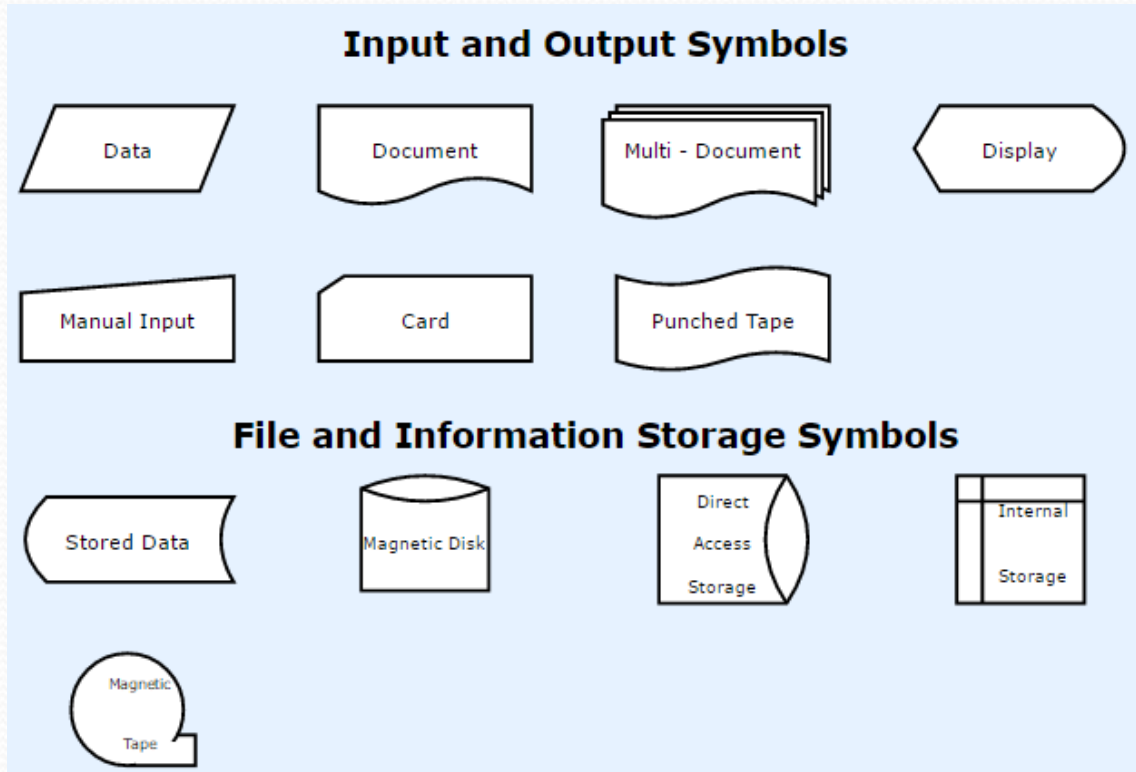
.Akış şemaları çeşitli alanlardaki işlem ve uygulamaların yönetilmesi, belgelendirilmesi, tasarlanması ve çözümlenmesinde kullanılır.

- Süreç akışını belgelemek için oluşturulan ilk metod olan "akış süreç şeması" Frank Gilbreth tarafından ASME (Amerikalı Mekanik Mühendisleri Topluluğu) üyelerine 1921'de sunulmuştur.
- Gilbreth'in araçları endüstri mühendisliği uygulaması yerini edinmiştir.
- 1930'ların başlarında, endüstri mühendisi olana Allan H.Mogensen New York'ta iş dünyasında kullanılmak üzere bazı endüstriyel uygulamaların eğitimini vermeye başladı.

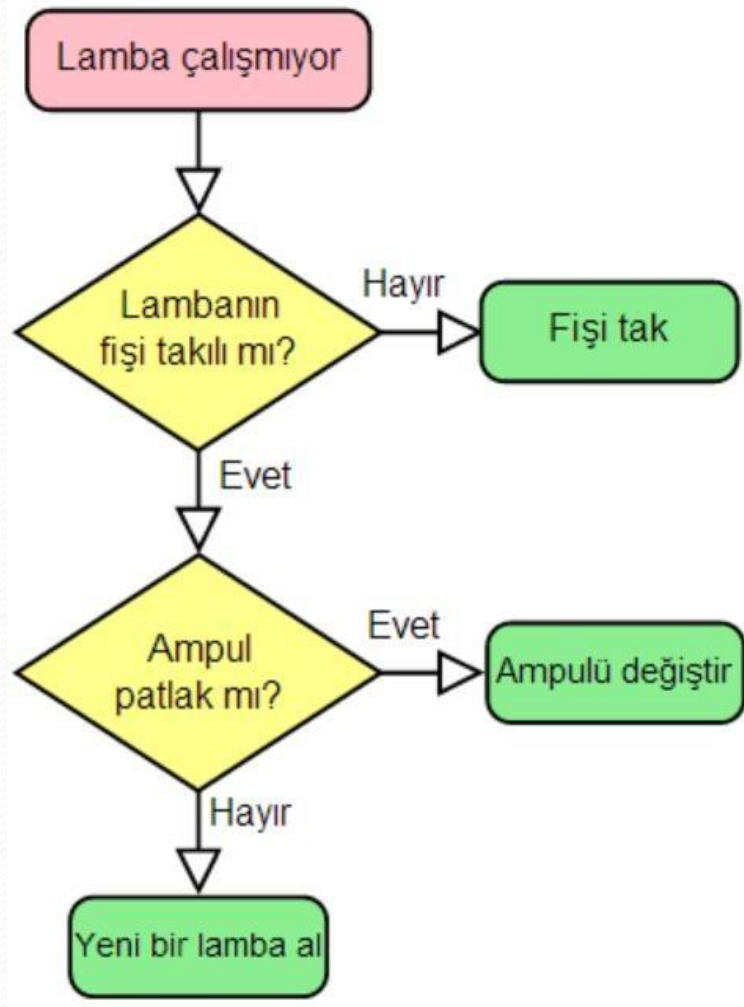
Akış Şeması Çiziminde kullanılan semboller



Akış Şeması Çiziminde kullanılan semboller

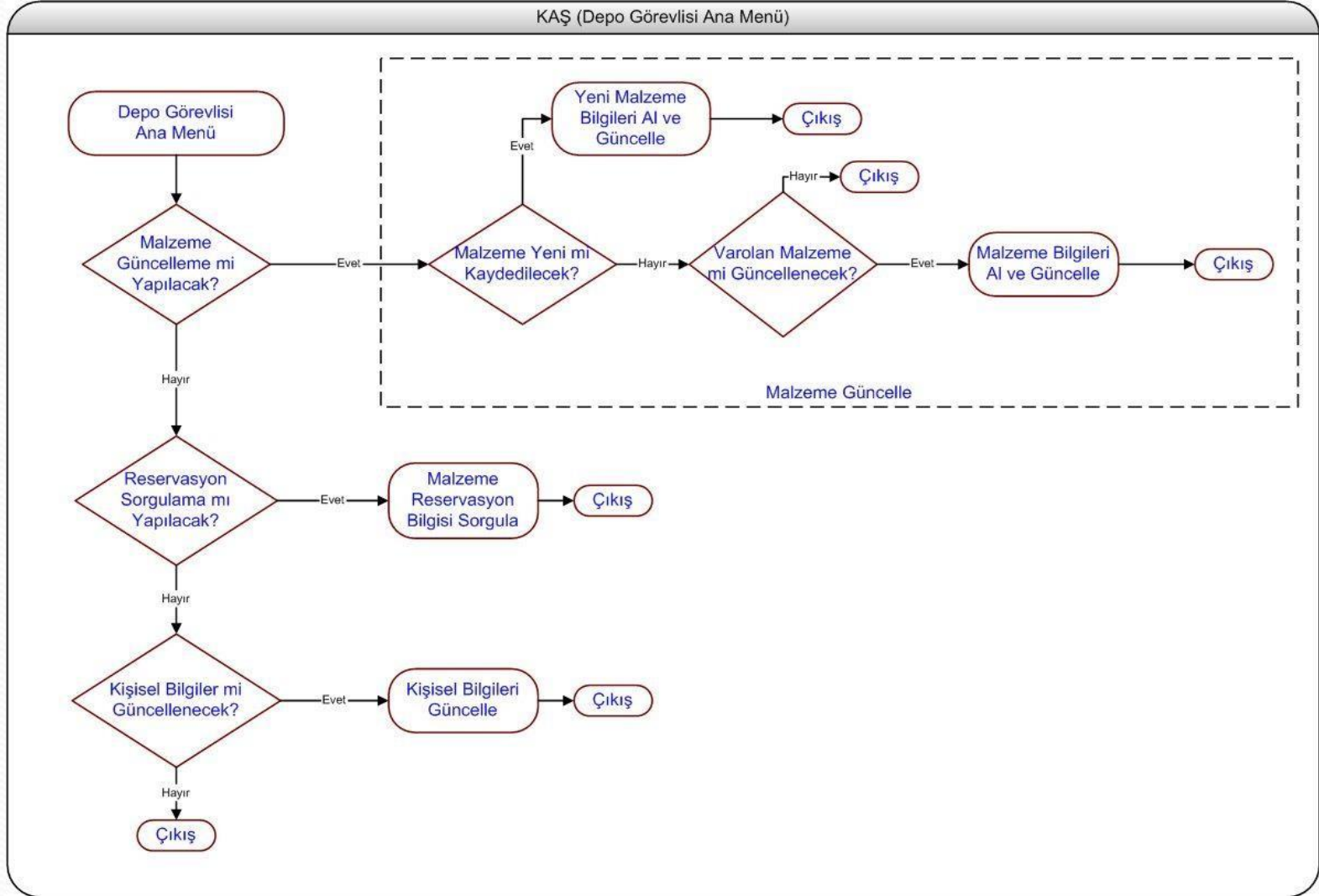


Akış Şeması



Akış şemaları sadece programlamada değil, bir çok alanda kullanılabilir.

Akış Şeması



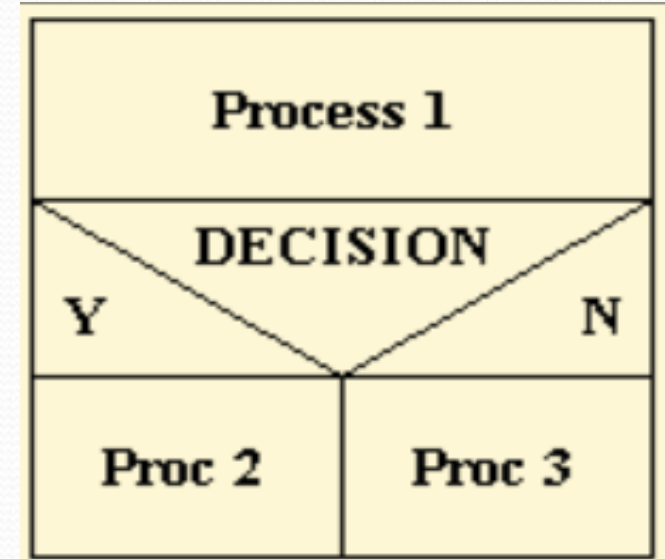
N-S (Nassi-Schneiderman) Şeması

Programın bir kutu içerisinde parçalara bölünüp ifade edilmesi.

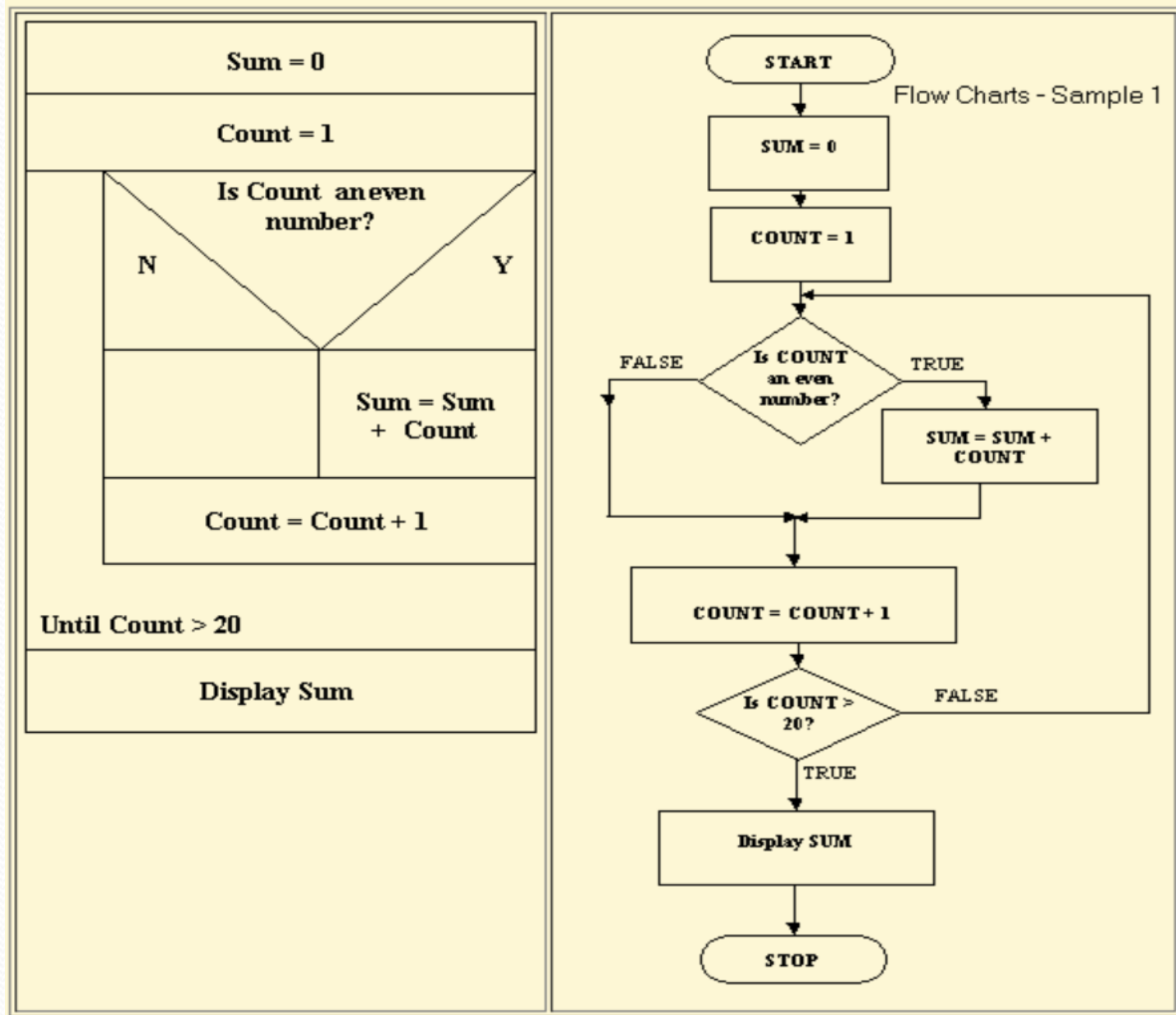
Bir karar (Decision) ardından bir süreç yandaki gibi bir diyagram ile bir algoritma gösterir.

Kararın her bir sonuç ile ilişkili için bir süreci vardır.

Karar doğru olarak değerlendirilirse, o zaman Proc 2 yürütülmektedir, eğer yanlış ise Proc 3 gerçekleştirilir.

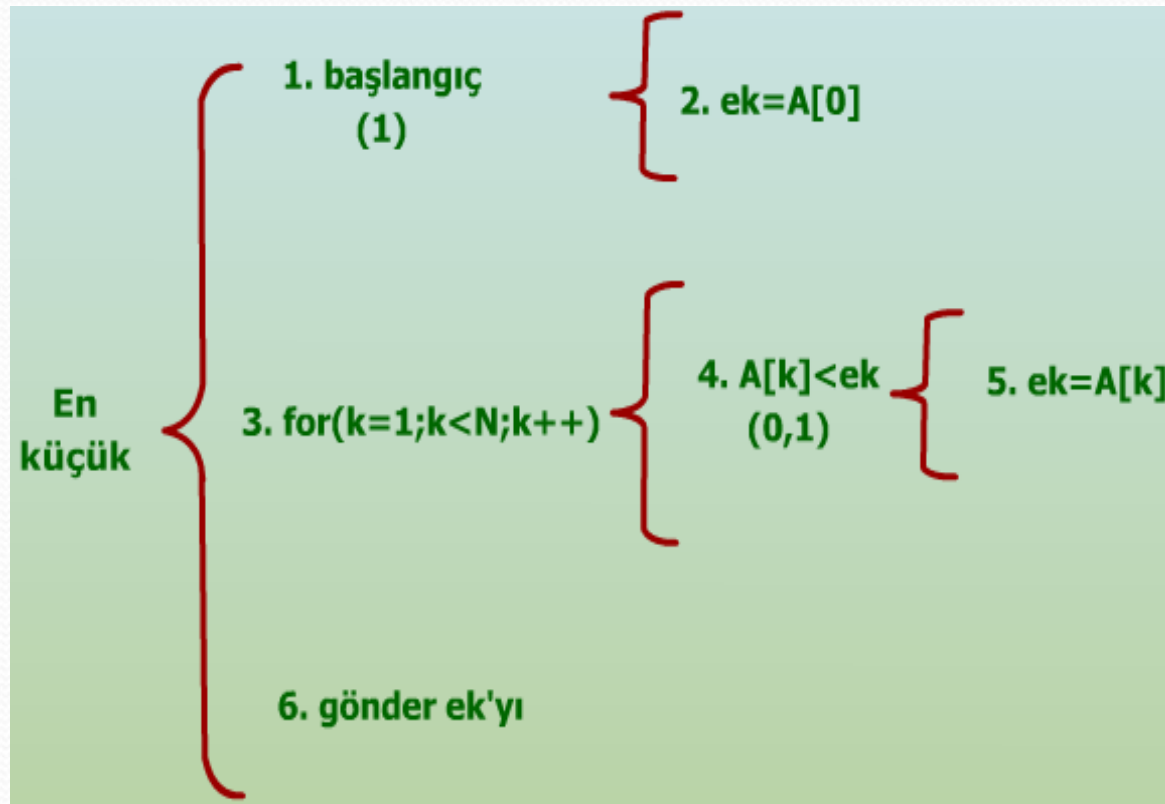


N-S (Nassi-Schneiderman) Şeması



W-O (Warnier-Orr) Diyagramı

.Programların küme parantezi şeklinde ifade edilmesi.



Algoritma Tarihi

- .Yaklaşık MÖ 1600 - Babilliler bilinen ilk çarpanlara ayırma ve kök bulma algoritmasını geliştirdiler.
- .Yaklaşık MÖ 300 - Öklid algoritması
- .Yaklaşık MÖ 200 - Eratosthenes Elemesi
- .263 - Gaussal eleme, Liu Hui tarafından tanımlandı.
- .813 ve 833 yılları arasında Harezmi doğrusal ve ikinci dereceden denklemleri çözmek için bir algoritma geliştirdi. Algoritma ismi bu kişinin adından türetilmiştir
- .1614 - John Napier logaritmayı kullanan hesaplamaları yapmak için bir metod geliştirdi
- .1671 - Newton-Raphson metodu, Isaac Newton tarafından geliştirildi.
- .1690 - Newton-Raphson methodu bağımsız olarak Joseph Raphson tarafından da bulundu
- .1805 - Cooley-Tukey algoritması Carl Friedrich Gauss tarafından biliniyordu
- .1926 - Boruvka algoritması
- .1934 - Delaunay üçgen bölümlenmesi Boris Delaunay tarafından geliştirildi

Bilgisayarda Önemli Algoritmaların Tarihi

- .Bilgisayar alanında yüzlerce algoritma mevcuttur. Aşağıda bazı algoritmalara yer verilmiştir.
- .1945 - Birleştirmeli sıralama John von Neumann tarafından geliştirildi
- .1952 - Huffman kodlaması David A. Huffman tarafından geliştirildi
- .1957 - Prim algoritması Robert Prim tarafından geliştirildi
- .1960 - Hızlı Sıralama C. A. R. Hoare tarafından geliştirildi
- .1973 - RSA şifreleme algoritması Clifford Cocks tarafından geliştirildi
- .1975 - Genetik algoritma John Holland tarafından popülerleştirildi
- .1978 - LZ78 algoritması LZ77 algoritmasından Abraham Lempel ve Jacob Ziv tarafından geliştirildi
- .1984 - LZW algoritması LZ78'den Terry Welch tarafından geliştirildi
- .1994 - Shor'un algoritması Peter Shor tarafından geliştirildi
- .2001 - LZMA sıkıştırma algoritması

Yeni algoritma geliřtirdim!!! Bunu bir ortama uygulamadıđınız sürece sevinmeyin.

.Algoritmalar, tek başlarına, genellikle patent verilebilir değildirler. Amerika Birleşik Devletleri'nde soyut kavramların, sayıların ve işaretlerin yalnızca basit yönlendirmelerinden oluşan bir iddia "süreç" oluşturmaz (USPTO 2006), ve bundan dolayı algoritmalar patent verilebilir değildir (Gottschalk v.Benson'da olduğu gibi).

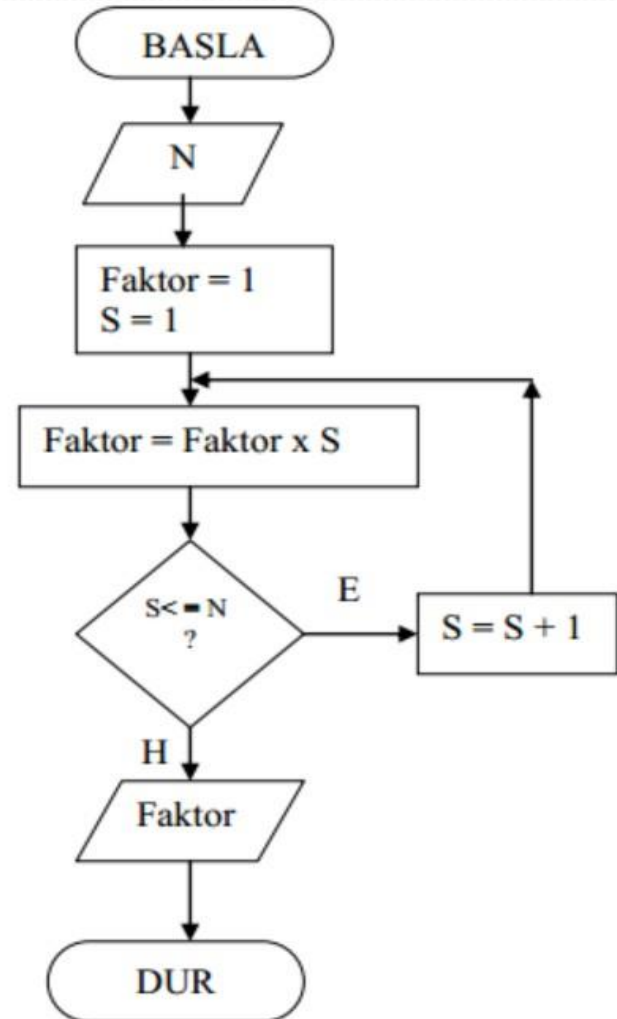
.Bununla birlikte, algoritmanın pratik uygulamaları zaman zaman patent verilebilirdir. Örneđin, Diamond v.Diehr'da, sentetik kauçuğun muhafaza edilmesine yardımcı olmak için kullanılan basit geri bildirim algoritmasının uygulaması patent verilebilir sayılmıştır.

.Yazılım patenti son derece tartışmalıdır ve algoritmaları içeren birçok eleştirilmiş patent vardır, özellikle veri sıkıştırma algoritmaları, Unisys' LZW patentinde olduğu gibi. Ek olarak, bazı kriptografik algoritmaların ihracat kısıtlamaları vardır.

Faktoriyel Hesabı

Algoritması

- 1) Başla
- 2) N'i oku
- 3) Faktor = 1
- 4) S = 1
- 5) Faktor = Faktor x S
- 6) Eğer $S \leq N$ ise $S = S + 1$ git 5
- 7) Yaz Faktor
- 8) Dur



Program Çalışma Hızı ve Bellek Gereksinimi

- Algoritma Analizi
- Yürütme Zamanı
- Zaman Karmaşıklığı
- Alan Maliyeti
- Alan Karmaşıklığı
- Algoritma Performansı
- Kıyaslama
- Örnekler

Tanımlar

.Programın Çalışma Hızı

- Belirlenen bir problemin çözümü için tasarlanan programın görevini yerine getirebilmesi için gerekli zaman ifadesini veren bir ifadedir.

.Bellek Gereksimini

- Programın çalışabilmesi için bellekte ihtiyaç duyulan alan.

Algoritma Analizi Kriterler

.Kod uzunluğu

- Daha kısa kod ile yazabilir miyim?

.Gereksiz kısımlar (bug, değişkenler)

- Gereksiz kısımları çıkarabilir miyim?

.Bellek gereksinimi

- Aynı işlemi daha az bellek kullanarak yapabilir miyim?

.Çalışma zamanı

- Aynı işlemi daha kısa sürede yapabilir miyim?

Temel Kavramlar

- .Yürütme Zamanı
- .Zaman Karmaşıklığı
- .Alan Maliyeti
- .Alan Karmaşıklığı
- .En iyi, Ortalama ve En Kötü Durumlar

Yürütme Zamanı (Running Time)

.Bir programın veya algoritmanın işlevini yerine getirebilmesi için temel kabul edilen işlemlerden kaç adet yürütülmesi gerektiğini veren bir bağıntıdır.

Örneğin n elemanlı bir küme için yürütme zamanı $T(n)$ şeklinde gösterilir.

.Temel hesap birimi olarak, programlama dilindeki deyimler seçilebildiği gibi döngü sayısı, toplama işlemi sayısı, atama sayısı, dosyaya erişme sayısı gibi işler de temel hesap birimi olarak seçilebilir.

Zaman Karmaşıklığı (Time Complexity)

.Algoritmanın zaman karmaşıklığı ise, belirli miktardaki giriş verisine karşılık, yapılan karşılaştırma, toplama, çıkartma, çarpma ve bölme işlemleri ile diğer basit işlemlerin sayısı olarak hesaplanır.

.Bilgisayarda hesaplamalar genellikle aşağıdaki işlemler yardımıyla gerçekleştirilir:

- Atama
- Karşılaştırma ($=$, $\neq$, $>$, $<$, $\leq$, $\geq$)
- Aritmetik işlemler ($+$, $-$, $\times$, $/$)
- Mantıksal operasyonlar

Zaman Karmaşıklığı

.Bu operatörlerin çalıştırılması ne kadar zaman alır?

- Genelde, atama çok hızlı yapılır ve diğer işlemler daha yavaştır. Çarpma ve bölme de toplama ve çıkarmadan daha yavaştır. Kayan noktalılarla işlem yapmak, tamsayılarla işlem yapmakta daha yavaştır.
- Çoğu algoritmada bu işlemlerden çok sayıda kullanılmaktadır ve hepsinin ne kadar zaman alacağını teker teker hesaplamak oldukça zor olacaktır. Bunun yerine çoğu algoritmada zaman karmaşıklılığı baskın olan tek bir işlem ile belirlenebilmektedir.