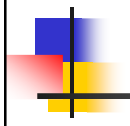


BM 307 Dosya Organizasyonu (File Organization)

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Konular

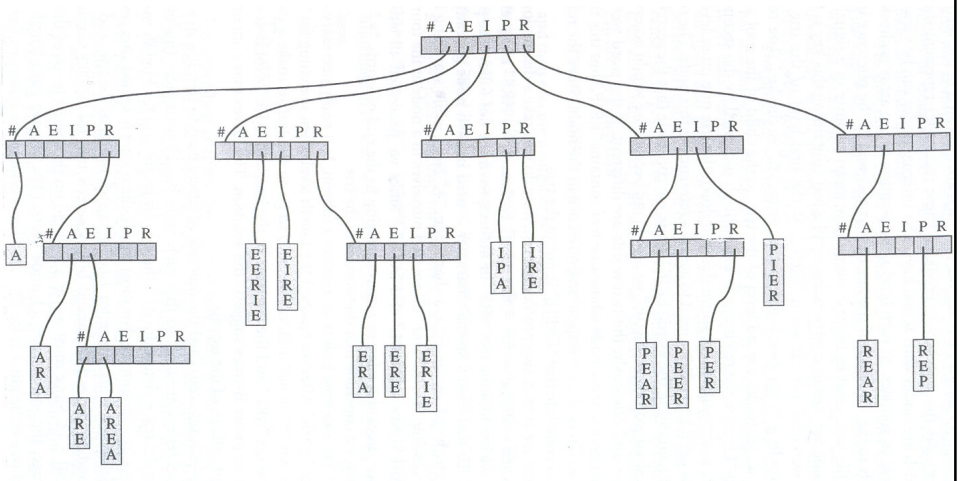
- Tries
  - Introduction
  - Trie Hashing
  - Değerlendirme
- Secondary Key Retrieval
  - K-d Trees
  - Değerlendirme

Tries
Introduction

- Trie aranan anahtarın bir kısmını ağaç üzerinde ilerlemek için kullanır.
- Re**trie**val kelimesinin bir ksımı alınarak oluşturulmuştur.
- Bir node, anahtardaki farklı karakter sayısı kadar elemana sahiptir.
- Ağaç üzerindeki tüm yapraklardaki bilgiler soldan sağa doğru alfabetik olarak yer alırlar.

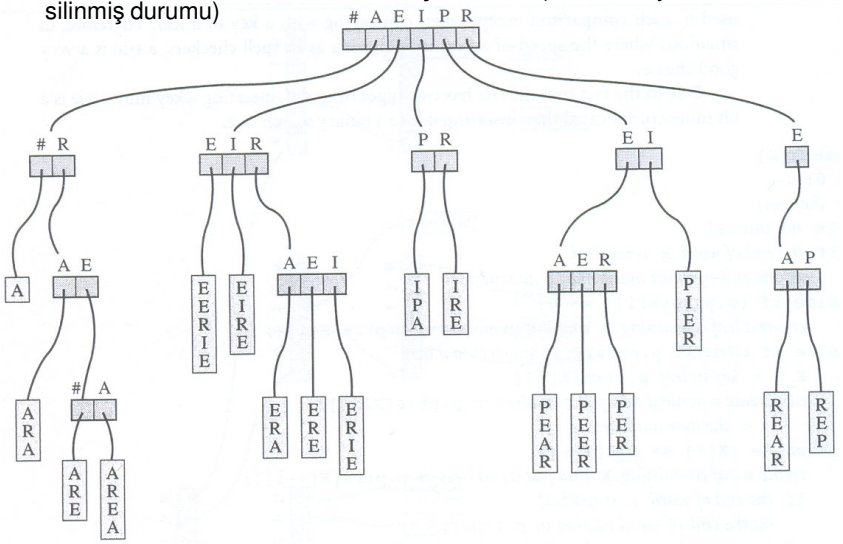
Tries
Introduction

- Örnek: A, E, I, P, R harflerinden oluşan bir trie



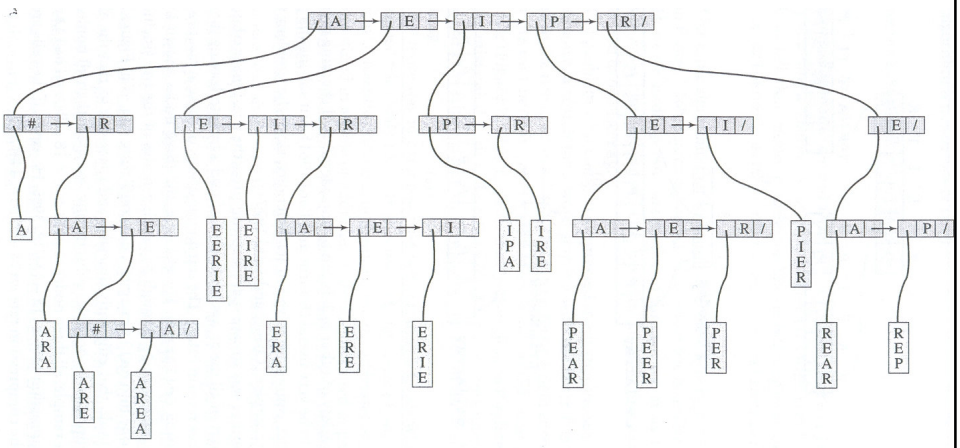
Tries
Introduction

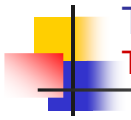
- Örnek: A, E, I, P, R harflerinden oluşan bir trie (kullanılmayan alanların silinmiş durumu)



Tries
Introduction

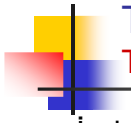
- Örnek: A, E, I, P, R harflerinden oluşan bir trie'ın binary tree olarak gösterimi





Tries
Trie Hashing

- Sonuçta elde edilen dosya sıralıdır.
- Ekstra işlem yapmadan doğrudan ve direkt erişim yapılabilir.
- Data sayfalarına bir indeks ile ulaşılır.
- İndeks primary memory'e yerleştirilirse tek erişimle bilgiye ulaşılır.
- Dosya yeniden organize edilmeden büyüyüp küçültülebilir.
- Hash işlemi için bir fonksiyona gerek duymaz.
- Data sayfalarında anahtarlar sıralıdır.
- Data sayfalarında kayıt sayısı belirlenenden fazla olduğunda overflow olur.
- Overflow olan sayfaiçin bir index node oluşturulur.
- Overflow olan sayfadaki kayıtlar ortadaki kayda göre iki sayfaya dağıtılır.



Tries
Trie Hashing

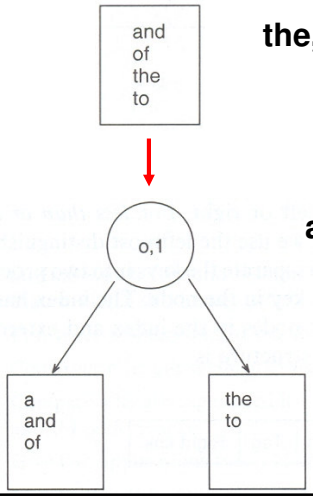
- İndex node ve sayfa yapısı farklıdır.
- İndeks node içerisinde 6 alan vardır

Left link	Tag	Character value	Position	Tag	Right link

- Left ve Right link, sol ve sağ node'ları gösterir.
- Tag, ilgili bağlantının index node veya data sayfası olduğunu gösterir.
- Character value, karşılaştırma için kullanılan karakteri gösterir
- Position, karakterin sırasını gösterir.

Tries
Trie Hashing

- Örnek 1: Sayfadaki kayıt sayısı 4 alınmıştır.



the, of, and, to insert edildi.

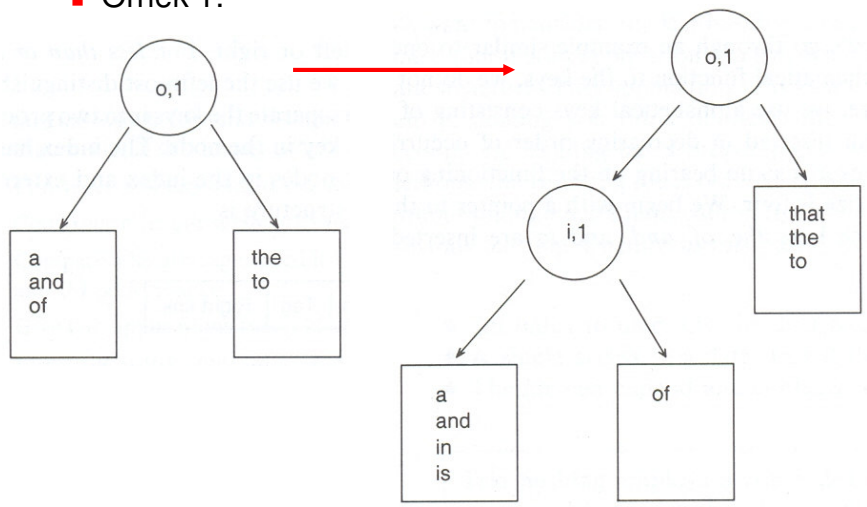
a insert edildi.

of anahtarının ilk harfine göre anahtarlar dağıtıldı.

Tries
Trie Hashing

- Örnek 1:

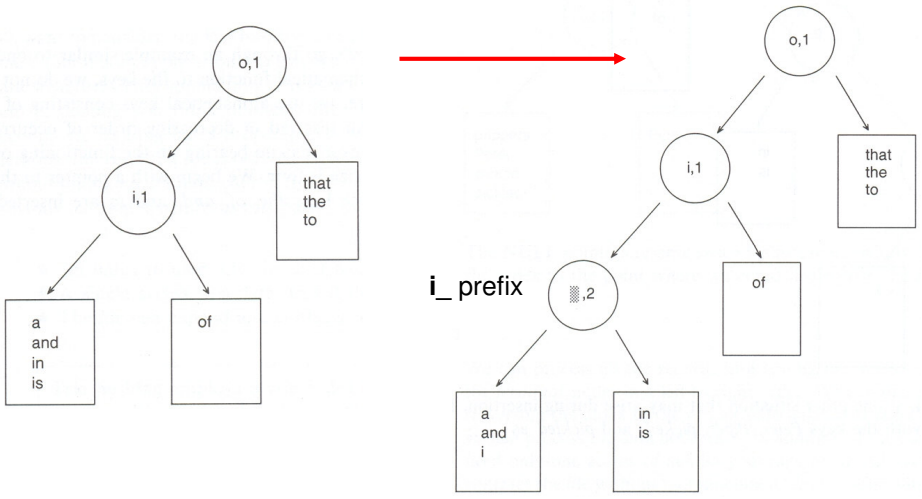
that, in, is insert edildi.



Tries
Trie Hashing

- Örnek 1:

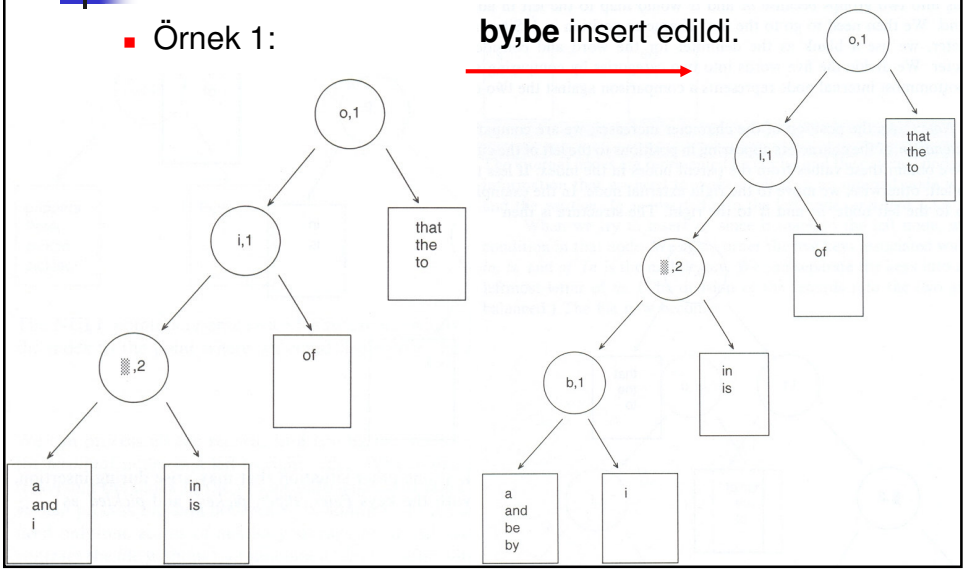
i insert edildi.



Tries
Trie Hashing

- Örnek 1:

by,be insert edildi.



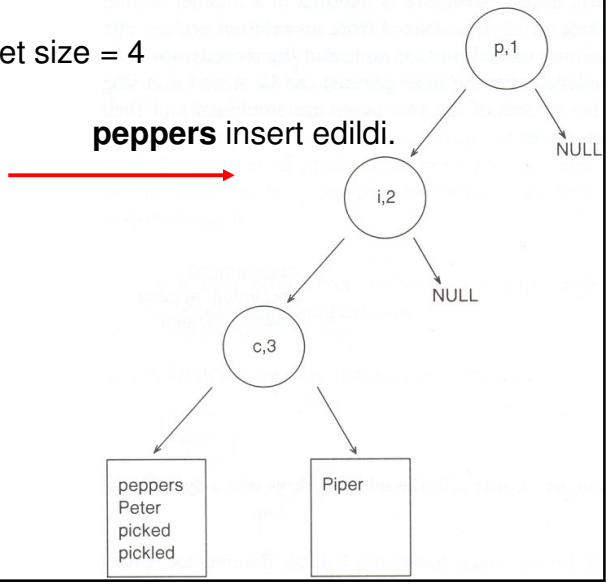
Tries
Trie Hashing

- Örnek 2: Bucket size = 4

Peter
picked
pickled
Piper

Peter, Piper, picked, pickled insert edildi

peppers insert edildi.



Tries
Trie Hashing

- **Değerlendirme**
  - Terminal node'larda soldan sağa doğru giderek tüm kayıtlar sıralı elde edilebilir.
  - Doğrudan bir kayda ağaçtaki indeks node'ları ile ulaşılabilir.
  - İndeks primary memory'e alınırsa auxiliary memory'e bir erişimle bilgiye ulaşılabilir.
  - Packing factor genellikle %70 seviyesindedir.
  - İndeks için extendible ve dynamic hashing metodlarına göre daha fazla alana ihtiyaç duyar.
  - Oluşturulan ağaç, B+ ağaçlardaki gibi dengeli değildir ve erişim probe sayısı anahtarın root node'a uzaklığına bağlıdır.
  - Silme işlemi dynamic hashing metodundaki gibi yapılır. Her silmeden sonra node'un aynı parent'a sahip diğer node ile birleştirilmesine çalışılır.

Secondary Key Retrieval
K-d Trees

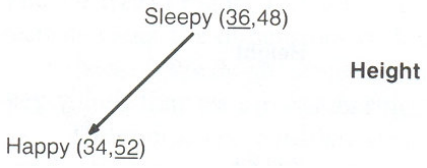
- K-d ağaçları ikincil anahtarlarda belirlenen bir aralığa bağlı arama yapar.
- K değeri arama yapılacak boyut (alan) sayısını ifade eder.
- K-d ağaçlarında root'dan yaprağa ilerledikçe arama alanı küçültülür.
- Karar verirken ≤ sola, ve > sağa ilerlemeyi gerektirir.
- K-d ağaçlarında her seviyede bir sonraki anahtara göre karşılaştırma yapılır.
- Sondaki anahtardan sonra tekrar baştaki anahtara dönülür.

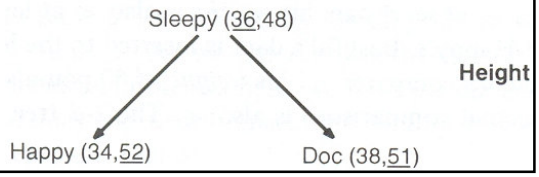
Secondary Key Retrieval
K-d Trees

- Örnek: Aşağıdaki tablodaki iki anahtarla 2-d ağacın oluşturulması.

TABLE 12.1 REPRESENTATIVE DATA RECORDS

Name	Height	Weight	Other data
Sleepy	36	48	
Happy	34	52	
Doc	38	51	
Dopey	37	54	
Grumpy	32	55	
Sneezy	35	46	
Bashful	33	50	
Ms. White	62	98	



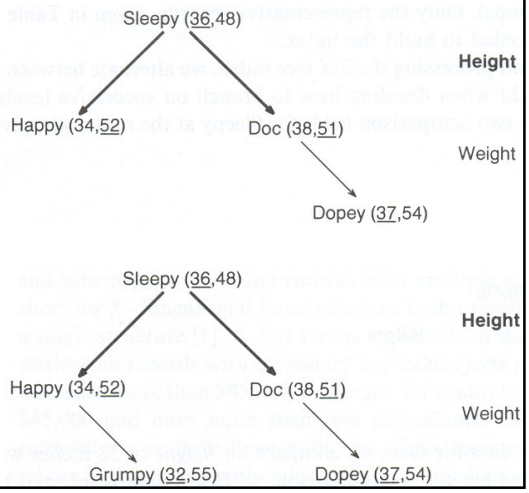


Secondary Key Retrieval
K-d Trees

- Örnek: Aşağıdaki tablodaki iki anahtarla 2-d ağacın oluşturulması.

TABLE 12.1 REPRESENTATIVE DATA RECORDS

Name	Height	Weight	Other data
Sleepy	36	48	
Happy	34	52	
Doc	38	51	
Dopey	37	54	
Grumpy	32	55	
Sneezy	35	46	
Bashful	33	50	
Ms. White	62	98	

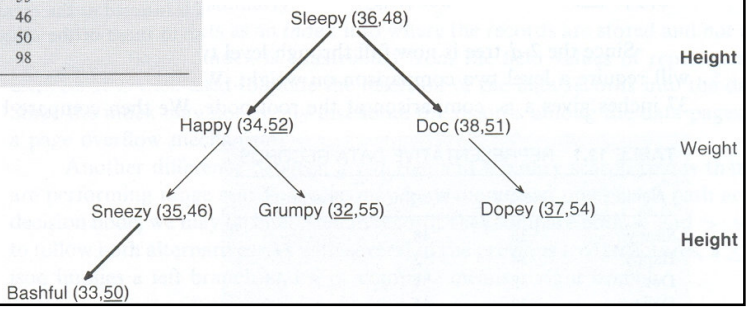


Secondary Key Retrieval
K-d Trees

- Örnek: Aşağıdaki tablodaki iki anahtarla 2-d ağacın oluşturulması.

TABLE 12.1 REPRESENTATIVE DATA RECORDS

Name	Height	Weight	Other data
Sleepy	36	48	
Happy	34	52	
Doc	38	51	
Dopey	37	54	
Grumpy	32	55	
Sneezy	35	46	
Bashful	33	50	
Ms. White	62	98	

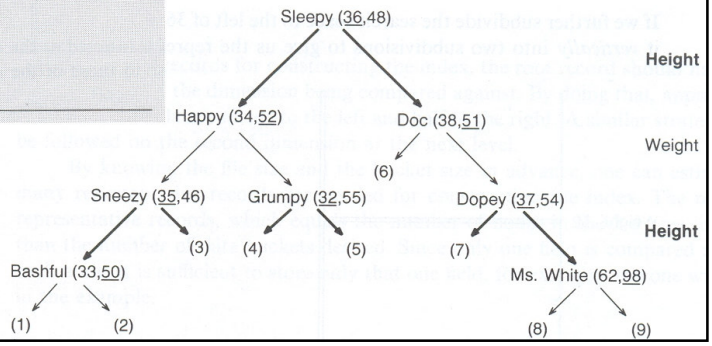


Secondary Key Retrieval
K-d Trees

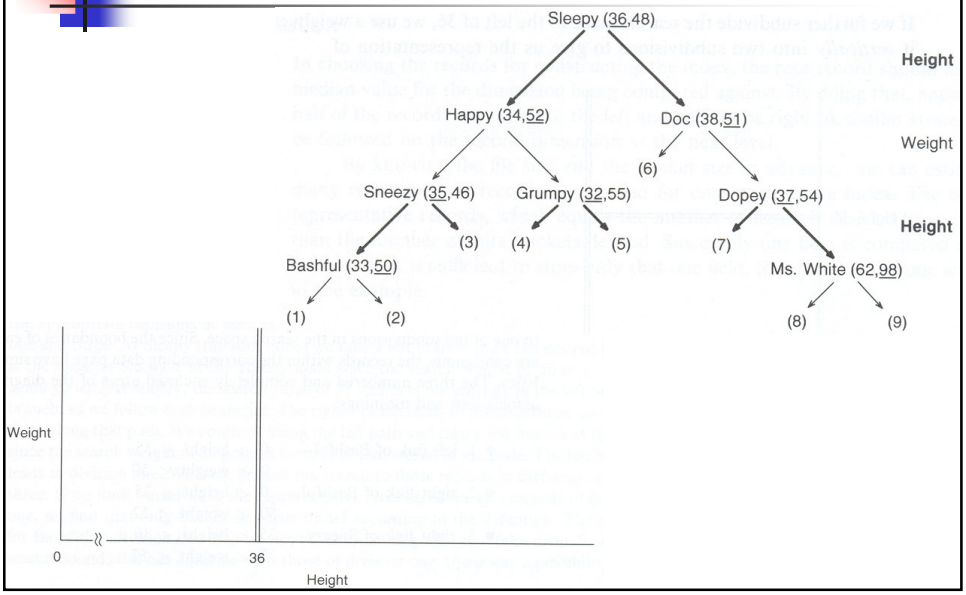
- Örnek: Aşağıdaki tablodaki iki anahtarla 2-d ağacın oluşturulması.

TABLE 12.1 REPRESENTATIVE DATA RECORDS

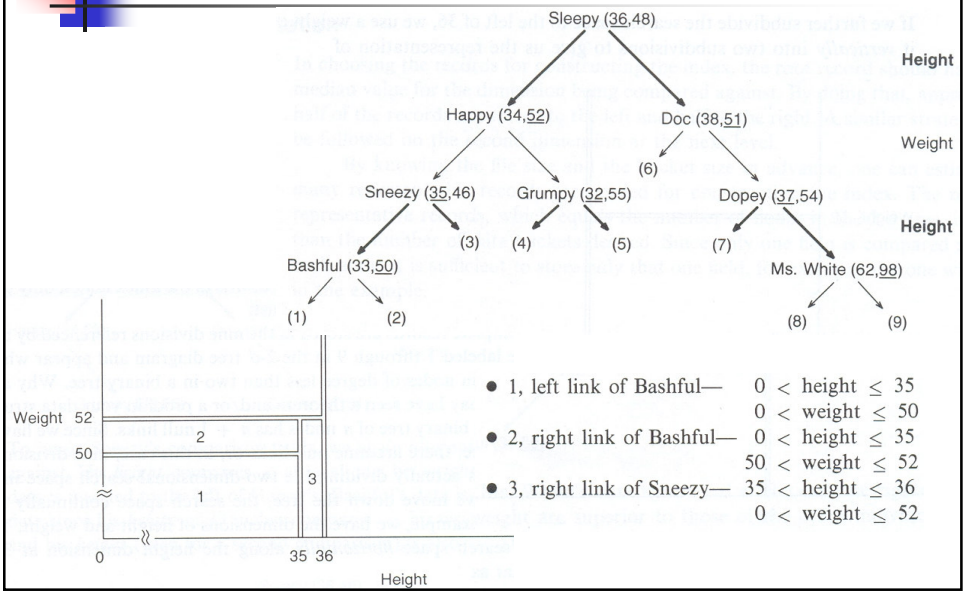
Name	Height	Weight	Other data
Sleepy	36	48	
Happy	34	52	
Doc	38	51	
Dopey	37	54	
Grumpy	32	55	
Sneezy	35	46	
Bashful	33	50	
Ms. White	62	98	



Secondary Key Retrieval
K-d Trees



Secondary Key Retrieval
K-d Trees



Secondary Key Retrieval
K-d Trees

- **Değerlendirme**
  - Root kaydın median değere yakın olması ağacın sol ve sağ yarısında kayıtların yarısının bulunmasını sağlar.
  - Daha sonraki seviyelerdede aynı özelliğin korunması ağacın dengeli olmasını sağlar.
  - K-d ağaçları, primary key ile aramanın yapılmadığı durumlarda kullanılır.
  - Bir aralığa göre ve birden fazla anahtar kullanılarak yapılan sorgulamalarda kullanılır.