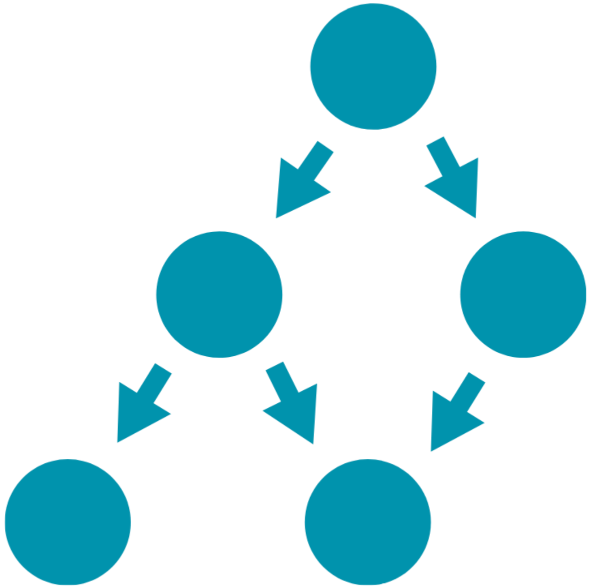
ZAFER CÖMERT
Öğretim Üyesi



VERİ YAPILARILARI VE ALGORİTMALAR

Giriş

Özyineleme Nedir?

What is Recursion?

- Kendini çağıran herhangi bir fonskiyon rekürsif (recursive) olarak adlandırılır.

- Özyinelemeli bir yöntem, daha küçük bir sorun üzerinde çalışmak için kendisinin bir kopyasını çağırarak bir sorunu çözer. Bu rekürsif adım (recursion step) olarak tanımlanır.

- Rekürsif adım çok daha fazla rekürsif çağrı ile sonuçlanır.

Özyineleme

- Durma koşulu olmalıdır.
- Küçük problemlerin daha küçük dizileri temel duruma (base case) yakınsamalıdır.
- Çoğu zaman iteratif kod yazmaktan daha kısa ve kolaydır.
- Benzer alt görevlerin kullanımında daha kullanışlı olurlar. Sıralama, arama ve gezinme problemleri bu duruma örnek olarak gösterilebilir.

Özyinelemeli Fonksiyonların Formatı

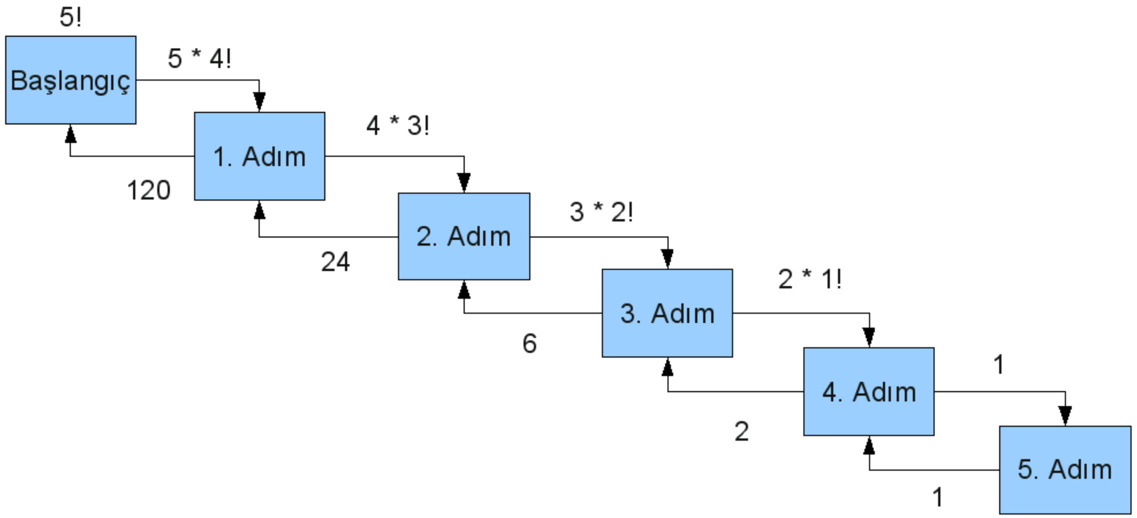
Format of Recursive Function

```
if(test for the base case)
        return some base case value
else if (test for another base case)
        return some another base case
value
else
        return (some work and then a
recursive call)
```

Faktöriyel

$$n! = \prod_{k=1}^{n} k = 1 * 2 * \cdots * (n-1) * n$$

$$n! = \begin{cases} 1 & n \leq 1 \\ n.(n-1)! & n > 1 \end{cases}$$



Recursion memory visualization

Fibonacci

$$F_n = F(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F(n-1) + F(n-2) & n > 1 \end{cases}$$

Endeks	0	1	2	3	4	5	6	7	…
Değeri	0	1	1	2	3	5	8	13	…

Permütasyon

$$P(n,r) = \binom{n}{n-r} = \frac{n!}{(n-r)!}$$

$Permütasyon\ (\{a,b,c\})$ ise;

1. $a\ Permütasyon(\{\ b,c\ \})$
   1.1. $b\ Permütasyon\ (\{c\}) \rightarrow abc$
   1.2. $c\ Permütasyon\ (\{b\}) \rightarrow acb$
2. $b\ Permütasyon(\{a,c\})$
   2.1. $a\ Permütasyon(\{c\}) \rightarrow bac$
   2.2. $c\ Permütasyon(\{a\}) \rightarrow bca$
3. $c\ Permütasyon(\{a,b\})$
   3.1. $a\ Permütasyon(\{b\}) \rightarrow cab$
   3.2. $b\ Permütasyon(\{a\}) \rightarrow cba$

Özyileme ve İterasyon

- Temel durum (base case) ulaşınca durur.
- Her rekürsif çağrı ekstra bellek alanı kullanır.
- Eğer sonsuz rekürsif çağrı yapılırsa; bellek taşma hatası alınır (`stack overflow`).
- Bazı problemlerin çözümü rekürsif olarak daha kolay ifade edilebilir.

- Bir koşulun yanlış olması durumunda durur.
- Her bir iterasyon ekstra bellek alanı gerektirmez.
- Ekstra bellek alanı gerektirmediğinden sonsuz döngüler sonsuza kadar devam eder.
- İteratif çözümler rekürsif çözümler kadar açık olmayabilir.

BTK AKADEMİ

Özyineleme için Notlar

- Rekürsif algoritmalar iki durum içermelidir.
  - Rekürsif durumlar ve temel durum.
- Her rekürsif fonksiyon durumu temel durumda durmalıdır.
- Genellikle iteratif çözümler rekürsif çözümlerden daha verimlidir.
  - Çünkü ekstra bellek alanı kullanmazlar.
- Bazı problemler en iyi rekürsif çözümler ile çözülebilirken bazıları için durum tam tersi olabilir.

Özyinelemeli Çağrılar için Örnekler

Example Algorithms of Recursion

Özyinelemeli Çağrılar

- Fibonacci series
- Factorial finding
- Merge sort, quick sort
- Binary search
- Tree traversals and many tree problems: InOrder, PreOrder, PostOrder
- Graph Traversals: DFS, BFS
- Dynamic Programming Examples
- Divide and Conquer Algorithms
- Towers of Hanoi
- Backtracking algorithms

Örnek: $T(n) = T\left(\frac{n}{2}\right) + c$, $T(1) = 1$ ve $n \geq 2$ koşuluna uygun olarak ilgili yinelemenin çözümünü yerine koyma (substitution) metodu ile çözünüz.

Örnek: $T(n) = T(\frac{n}{2}) + c$, $T(1) = 1$ ve $n \geq 2$ koşuluna uygun olarak ilgili yinelemenin çözümünü yerine koyma (substitution) metodu ile çözünüz.

$$T(n) = T(\tfrac{n}{2}) + c$$

$$T(2) = T(1) + c = 1 + c$$

$$T(4) = T(2) + c = (1+c) + c = 1 + 2c$$

$$T(8) = T(4) + c = (1+2c) + c = 1 + 3c$$

$$\vdots \qquad \vdots$$

$$T(2^k) = 1 + k.c$$

$$2^k = n$$

$$\log_2 2^k = \log_2 n$$

$$k = \log_2 n$$

$$T(n) = 1 + \log n . c$$

$$\Theta(\log n)$$

Örnek: Hanoi kulesi

$$T(n) = 2T(n-1) + 1$$

ve $T(0) = 0$ problemini yerine koyma metodu ile çözünüz.

Örnek: Hanoi kulesi

$T(n) = 2T(n-1) + 1$ ve $T(0)=0$ problemini yerine koyma metodu ile çözünüz.

$T(0) = 0$

$T(1) = 2T(0)+1 = 0+1 = 1$

$T(2) = 2T(1)+1 = 2.1+1$

$T(3) = 2T(2)+1 = 2.(2.1+1)+1 = 2^2+2+1$

$T(4) = 2T(3)+1 = 2.(2^2+2+1)+1 = 2^3+2^2+2+1$

$\vdots$

$f(n) = \sum_{0 \le i \le n} 2^i = \dfrac{(2^{n+1}-1)}{2-1} = 2^{n+1}-1$ olur.

Örnek: $T(n) = 2T\left(\frac{n}{2}\right) + n,$ $n > 1$, $T(1) = 1$ için iterasyon yöntemiyle çözünüz.

Örnek: $T(n) = 2T\left(\frac{n}{2}\right) + n$, $n>1$, $T(1)=1$ için iterasyon yöntemiyle çözünüz.

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + \frac{n}{2}$$

$$T(n) = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$T(n) = 2^2 . T\left(\frac{n}{4}\right) + 2n$$

$$T\left(\frac{n}{4}\right) = 2 . T\left(\frac{n}{8}\right) + \frac{n}{4}$$

$$T(n) = 2^2 \left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$$

$$T(n) = 2^3 . T\left(\frac{n}{8}\right) + 3n$$

$$\vdots$$

$$T(n) = 2^k . T\left(\frac{n}{2^k}\right) + k.n$$

$$\frac{n}{2^k} = 1 \Rightarrow n = 2^k$$

$$\log_2 n = \log_2 2^k$$

$$\log_2 n = k$$

$$T(n) = 2^k . T\left(\frac{n}{2^k}\right) + k.n$$

$$= n . T(1) + \log_2 . n$$

$$= n + n . \log n$$

$$T(n) = \Theta(n \log n)$$

Böl ve Yönet (Divide and conquer)

- Quick sort
- Merge sort
- Binary search

gibi ifadelerim böl ve yönet algoritma yaklaşımını benimser.

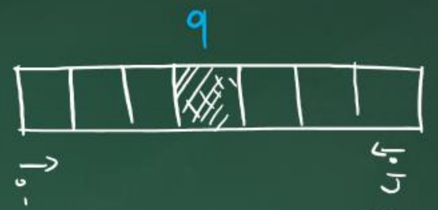
$P \to P_1, P_2, \ldots, P_n$

Durum 1:
$\exists i$, P_i yeterince basit hale gelirse çözümü yapılabilir.

Durum 2:
$\exists j$, P_j çözümü P_k bağımlı olmayacak $(k \neq j)$
$k = 1, 2, \ldots, n$

$Qs(p, i, j)$

$$\left\{ \begin{array}{l} p \leftarrow D[i] \\ D[i] \leftrightarrow p \end{array} \right. \; - \; O(n)$$



$p[i] \geq p \qquad p[j] \leq p$

$p[i] \leftrightarrow p[j]$

$Qs(p, i, q-1) \to T\left(\frac{n}{b}\right)$

$Qs(p, q+1, j) \to T\left(\frac{n}{b}\right)$

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

$$= a^k \, T(1) + \sum_{i=0}^{k-1} a^i f\left(\frac{n}{b^i}\right)$$

$\frac{n}{b^k} = 1 \Rightarrow n = b^k \qquad k = \log_b n$

$a^{\log_b n} \Rightarrow n^{\log_b a}$ ve $T(1) = \Theta(1)$

$$T(n) = \underbrace{n^{\log_b a} \cdot \Theta(1)}_{\text{Baskın terim?}} + \underbrace{\sum_{i=0}^{\log_b n - 1} f\left(\frac{n}{b^i}\right)}_{\text{Baskın Terim?}}$$

Master Theorem for Divide and Conquer

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

divide — half-size of problem

additional work for merging

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \Theta(n^k \log^p n)$$

where $a \geq 1$, $b > 1$, $k \geq 0$

$p \rightarrow$ is a real number

1) If $a > b^k$ then;

$T(n) = \Theta(n^{\log_b a} \cdot \log^{p+1} n)$

2) If $a = b^k$

a. If $p > -1$ $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

b. If $p = -1$ $T(n) = \Theta(n^{\log_b a} \log\log n)$

c. If $p < -1$ $T(n) = \Theta(n^{\log_b a})$

3) If $a < b^k$

a. If $p \geq 0$ $T(n) = \Theta(n^k \log^p n)$

b. If $p < 0$ $T(n) = O(n^k)$

Örnek : $T(n) = 9T\left(\frac{n}{3}\right) + n$

Örnek: $T(n) = 9\,T\left(\frac{n}{3}\right) + n$

$a = 9$

$b = 3$

$\boxed{f(n) = n}$ $\leftrightarrow$ karşılaştırma yapılır.

$n^{\log_b a} \Rightarrow n^{\log_3 3^2} = \boxed{\Theta(n^2)}$

$f(n) = O(n^{\log_3 9 - \epsilon})$ olur ve $\epsilon = 1$

olduğu için 2. kural geçerlidir.

ŞABLON $\rightarrow$ $T(n) = a\,T\left(\frac{n}{b}\right) + f(n)$

$f(n) = O(n^{\log_b a - \epsilon})$ için

$T(n) = \Theta(n^{\log_b a})$

Çözüm $\rightarrow$ $T(n) = \Theta(n^2)$

Örnek: $2T\left(\frac{n}{2}\right) + n$

Örnek: $2T\left(\frac{n}{2}\right)+n$

ŞABLON $\rightarrow a.T\left(\frac{n}{b}\right)+f(n)$

$a=2$

$b=2$

$f(n)=\boxed{n}$

Karşılaştır!

$n^{\log_b a} = n^{\log_2 2} = \boxed{\Theta(n^1)}$

CASE-2

$f(n)=O(n^{\log_b a})$ için $T(n)=\Theta(n^{\log_b a}\log n)$

$T(n)=\Theta(n^1 . \log n)$

$=\Theta(n\log n)$

Örnek: $T(n) = 4T\left(\frac{n}{2}\right) + n^2$

Örnek: $T(n) = 4T\left(\frac{n}{2}\right) + n^2$

ŞABLON $\rightarrow$ $a \cdot T\left(\frac{n}{b}\right) + f(n)$

$a = 4$

$b = 2$

$f(n) = n^2$

$n^{\log_b a} \leftrightarrow f(n)$

$n^{\log_2 2^2} \leftrightarrow n^2$

$n^2 \leftrightarrow n^2$ (Durum 2)

$T(n) = \Theta(n^2 \cdot \log n)$

$T(n) = \Theta(n^{\log_b a} \cdot \log n)$

$f(n) = \Theta(n \cdot \log_b^a)$

Örnek: $T(n) = 6T\left(\frac{n}{3}\right) + n^2 \log n$

Örnek: $T(n) = 6T\left(\frac{n}{3}\right) + n^2 \log n$

ŞABLON $\rightarrow$ $a \cdot T\left(\frac{n}{b}\right) + f(n)$

$a = 6$

$b = 3$

$f(n) = n^2 \log n$

$n^{\log_b a} \longleftrightarrow f(n)$

$n^{\log_3 6} \approx 1.5 \longleftrightarrow n^2 \cdot \log n$ (Durum 3)

$f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ $\epsilon > 0$

$a f\left(\frac{n}{b}\right) \leq c \cdot f(n)$ $c < 1$

$T(n) = \Theta(n^2 \log n)$

Örnek: $T(n) = 16T\left(\frac{n}{4}\right) + n!$

Örnek: $T(n) = 16\,T\left(\frac{n}{4}\right) + n!$

ŞABLON

$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

$a = 16$ $\quad a \geq 1$

$b = 4$ $\quad b > 1$

$f(n) = n!$

$n^{\log_b a} \leftrightarrow f(n)$

$n^{\log_4 4^2} \leftrightarrow n!$

$n^2 \leftrightarrow n!$

$a < b$

DURUM - 3

$f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ ve $a\,f\left(\frac{n}{b}\right) < c \cdot f(n)$

$= \Theta(f(n))$

$T(n) = \Theta(f(n))$

$= \Theta(n!)$

Veri Yapıları ve Algoritmalar

ZAFER CÖMERT
Öğretim Üyesi