# Veza između rekurentnih relacija, dinamičkog programiranja i pristupa „podeli pa vladaj“

## Sadržaj


**1 Uvod** **2**

**2 Rekurentne relacije** **2**
2.1 Primer . . . . . 2

**3 Pristup „podeli pa vladaj“** **2**
3.1 Koraci pristupa . . . . . 2
3.2 Rekurentne relacije u „podeli pa vladaj“ . . . . . 2

**4 Dinamičko programiranje** **2**
4.1 Koraci dinamičkog programiranja . . . . . 3
4.2 Rekurentne relacije u dinamičkom programiranju . . . . . 3

**5 Povezanost između koncepata** **3**

**6 Analiza određenih algoritama** **3**
6.1 Merge Sort (Spajanje nizova) . . . . . 3
6.2 Binary Search (Binarna pretraga) . . . . . 4
6.3 Faktorijel . . . . . 4
6.4 Fibonacijev niz . . . . . 4
6.5 Tower of Hanoi (Hanojeva kula) . . . . . 4

**7 Veza između rekurentnih relacija i ovih algoritama** **4**

**8 Dinamičko programiranje vs. „Podeli pa vladaj“** **5**
8.1 Sličnosti . . . . . 5
8.2 Razlike . . . . . 5

**9 Zaključak** **5**

# 1 Uvod

Rekurentne relacije, dinamičko programiranje i pristup „podeli pa vladaj“ su fundamentalni koncepti u računarstvu koji su međusobno povezani i često se koriste u dizajniranju efikasnih algoritama.

# 2 Rekurentne relacije

Rekurentne relacije su jednačine koje definišu funkciju u smislu njenih vrednosti na manjim ili prethodnim unosima. U kontekstu algoritama, rekurentne relacije često opisuju vremensku složenost rekurzivnih procedura.

## 2.1 Primer

Vremenska složenost funkcije koja računa $n$-ti Fibonacijev broj može se definisati kao:

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

gde je $c$ konstanta koja predstavlja osnovne operacije.

# 3 Pristup „podeli pa vladaj“

„Podeli pa vladaj“ je algoritamski pristup koji rešava problem tako što ga deli na manje podprograme, rešava svaki od njih rekurzivno i zatim kombinuje rezultate da bi se dobilo konačno rešenje.

## 3.1 Koraci pristupa

1. **Podela (Divide):** Problem se deli na manje podprograme.
2. **Osvajanje (Conquer):** Podprogrami se rešavaju rekurzivno.
3. **Kombinovanje (Combine):** Rezultati podprograma se kombinuju da bi se dobilo rešenje originalnog problema.

## 3.2 Rekurentne relacije u „podeli pa vladaj“

Vremenska složenost ovih algoritama često se izražava rekurentnim relacijama, jer se problem rešava na osnovu rešenja svojih podproblema.

# 4 Dinamičko programiranje

Dinamičko programiranje je tehnika koja se koristi za rešavanje problema koji mogu biti podeljeni na preklapajuće podprograme. Umesto da rešava iste podprograme više puta, dinamičko programiranje pamti rezultate podprograma i koristi ih po potrebi.

### 4.1 Koraci dinamičkog programiranja

1. **Optimalna podstruktura:** Problem se može rešiti optimalno na osnovu optimalnih rešenja njegovih podprograma.
2. **Preklapanje podprograma:** Podprogrami se ponavljaju i njihovi rezultati se mogu sačuvati (memoizacija ili tabulacija).
3. **Konstrukcija rešenja:** Kombinovanjem rezultata podprograma dobijamo konačno rešenje.

### 4.2 Rekurentne relacije u dinamičkom programiranju

One definišu način na koji se rešenje problema gradi na osnovu rešenja manjih podproblema.

## 5 Povezanost između koncepata

- **Rekurentne relacije** su osnova za oba pristupa:
  - U „podeli pa vladaj“, one opisuju kako se problem deli i kombinuje.
  - U dinamičkom programiranju, one definišu kako se rešenje gradi iz podprograma.
- **Razlika između pristupa:**
  - **Podeli pa vladaj**: Fokusira se na probleme koji se mogu podeliti na nezavisne podprograme.
  - **Dinamičko programiranje**: Bavi se problemima sa preklapajućim podprogramima.

## 6 Analiza određenih algoritama

### 6.1 Merge Sort (Spajanje nizova)

- **Pristup**: Podeli pa vladaj.
- **Kako funkcioniše**:
  - **Podela**: Niz se deli na dve polovine.
  - **Osvajanje**: Svaka polovina se sortira rekurzivno.
  - **Kombinovanje**: Dve sortirane polovine se spajaju.
- **Rekurentna relacija**:

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

## 6.2 Binary Search (Binarna pretraga)

- **Pristup**: Podeli pa vladaj.
- **Kako funkcioniše**:
  - Upoređuje se srednji element sa traženom vrednošću.
  - Pretraga se nastavlja u odgovarajućoj polovini.
- **Rekurentna relacija**:
$$T(n) = T\left(\frac{n}{2}\right) + O(1)$$

## 6.3 Faktorijel

- **Pristup**: Jednostavna rekurzija.
- **Kako funkcioniše**:
$$n! = n \times (n-1)!$$
- **Rekurentna relacija**:
$$T(n) = T(n-1) + O(1)$$

## 6.4 Fibonacijev niz

- **Pristup**: Rekurzija ili dinamičko programiranje.
- **Kako funkcioniše**:
$$F(n) = F(n-1) + F(n-2)$$
- **Rekurentna relacija**:
$$T(n) = T(n-1) + T(n-2) + O(1)$$

## 6.5 Tower of Hanoi (Hanojeva kula)

- **Pristup**: Rekurzija i „podeli pa vladaj“.
- **Kako funkcioniše**:
  - Premeštanje $n-1$ diskova na pomoćni stub.
  - Premeštanje najvećeg diska na cilj.
  - Premeštanje $n-1$ diskova sa pomoćnog na cilj.
- **Rekurentna relacija**:
$$T(n) = 2T(n-1) + O(1)$$

# 7 Veza između rekurentnih relacija i ovih algoritama

Rekurentne relacije omogućavaju analizu vremenske složenosti rekurzivnih algoritama i pomažu u razumevanju njihove strukture.

# 8 Dinamičko programiranje vs. „Podeli pa vladaj“

## 8.1 Sličnosti

- Koriste rekurziju i rekurentne relacije.
- Deli problem na podprograme.

## 8.2 Razlike

- **Preklapanje podprograma**:
  - **Dinamičko programiranje**: Podprogrami se preklapaju; rezultati se pamte.
  - **Podeli pa vladaj**: Podprogrami su nezavisni; nema potrebe za čuvanjem rezultata.

# 9 Zaključak

Razumevanje rekurentnih relacija, dinamičkog programiranja i pristupa „podeli pa vladaj“ je ključno za dizajniranje efikasnih algoritama i analizu njihove vremenske složenosti. Ovi koncepti su međusobno povezani i pružaju moćne alate za rešavanje složenih problema.