Dizajn i analiza algoritama

Lekcija 7

leto 2019/2020

Prof. dr Branimir M. Trenkić

Rekurzivni algoritmi - Uvod

- ***Pristup*** koji se najednostavnije može objasniti – ***do rešenja*** se dolazi u postupku ***"od opšteg ka pojedinačnom"***:

1. Dati ***problem*** se najpre ***podeli*** ***u*** više manjih ***potproblema***
2. Zatim se ***nezavisno*** ***rešava svaki*** od tih potproblema
3. ***Kombinovanjem*** njihovih rešenja se dolazi ***do rešenja polaznog problema***

Rekurzivni algoritmi - Uvod

- ***1. korak***:
  1. ***Manji potproblemi*** moraju biti ***slični polaznom*** problemu (***logički isti*** - ***samo jednostavniji!***)
  2. ***Postupak deljenja*** u manje potprobleme mora:
     - Završiti se posle ***konačno mnogo koraka***
     - Na kraju dobiti ***trivijalni potproblem*** – čije se rešenje može ***neposredno dobiti bez daljeg deljenja***
- **Dakle,**
- ***Rekurzivni pristup*** u rešavanju problema se zasniva na ***višestrukoj*** (rekurzivnoj) ***primeni istog postupka***

Rekurzivni algoritmi - Uvod

- ***Složeni problemi*** – ***Rekurzivni način*** rešavanja
- Rekurzivni način rešavanje problema u programiranju je ***vrlo važan i <u>efikasan metod</u>*** za rešavanje takvih problema
- Takvi problemi se mogu ***prirodnije i <u>jednostavnije rešiti</u>*** primenom rekurzije nego iterativnim putem

Rekurzivni algoritmi - Uvod

- Rekurzivno rešavanje problema – zahteva ***drugačiji pristup*** u načinu programiranja
- ***"rekurzivno razmišljanje"***
- Ilustrovaćemo ga ***kroz više*** reprezentativnih ***primera***
- ***Analiza*** rekurzivnih algoritama
  - Ne sastoji se od pukog ***prebrojavanja*** jediničnih ***instrukcija***
  - Zasniva se na rešavanju tzv. ***rekurentne jednačine***

Rekurzivni vs Iterativni

- ***Iterativni algoritam***: ***ponavljanje*** nekog ***postupka*** više puta
  1. Postupak koji se ponavlja navodi se ***u formi tela petlje***
  2. Broj ponavljanje se kontroliše ***izlaznim uslovom petlje***
- ***Rekurzivni algoritam***: ***ponavljanje*** nekog ***postupka*** više puta
  1. Postupak koji se ponavlja je ***sam algoritam***
  2. Broj ponavljanje se kontroliše ***naredbom grananja*** u rekurzivnom algoritmu

Rekurzivni vs Iterativni

- ***Svaki problem*** koji se može rešiti ***rekurzivno*** može se rešiti i ***iterativno***, kao i obrnuto
- ***Napomena***:
- ***Postupak*** koji se ***ponavlja u rekurzivnom*** algoritmu – ***ne mora biti identičan*** postupku koji se ponavlja u iterativnom algoritmu

Rekurzivni vs Iterativni

- ***Primer***: Izračunavanje stepena x^n, gde je x neki ***realan broj*** različit od nule i $n \geq 0$ ceo broj
- ***Iterativno rešenje*** za x^n:
- ***Postupak*** - sastoji se od ***uzastopnog množenja*** broja x sa ***parcijalno izračunatim stepenima*** x^i za $i = 0,1,...,n-1$
- Algoritam – ***power1*** koristi ***for- petlju*** radi realizacije ovog postupka

Rekurzivni vs Iterativni

- ***Algoritam – power1:***

```
// Ulaz:  realan broj x, ceo broj n ≥ 0
// Izlaz: broj x^n

algorithm power(x, n)

   y = 1;
   for i = 1 to n do
      y = x * y;

   return y;
```

Rekurzivni vs Iterativni

- ***Rekurzivno rešenje*** za x^n:
- ***Postupak*** –
- Uočimo da je za stepen ***n = 0*** ***rezultat*** je uvek ***1***, tj. $x^0 = 1$
- Za stepen ***n > 0***, ***rezultat*** dobijamo tako što ***x*** pomnožimo sa ***(n-1)-im stepenom*** broja ***x***,

$$x^n = \underbrace{x \cdot x \cdot \cdots \cdot x}_{n} \quad = \textbf{power(x,n)}$$

$$= x \cdot \underbrace{x \cdot \cdots \cdot x}_{n-1} \quad = \textbf{power(x,n-1)}$$

$$= x \cdot x^{n-1} \qquad \ldots\ldots\ldots\ldots\ldots$$

- Algoritam – ***power***

Rekurzivni vs Iterativni

- ***Rekurzivno rešenje*** za x^n:
- Izračunavanje se ***podeli u dva slučaja***:
- ***Bazni slučaj***: ako je ***n = 0***, tada je ***power(x,0) = 1***
- ***Opšti slučaj***: ako je ***n ≠ 0***, onda je

  power(x,n) = x*power (x, n-1)
- ***Rekurzivna definicija*** ***power***:

$$power(x,n) = \begin{cases} 1, & \text{ako je } n = 0 \\ x \cdot power(x, n-1), & \text{ako je } n \neq 0 \end{cases}$$

Rekurzivni vs Iterativni

- ***Algoritam – power:***

```
// Ulaz:  realan broj x, ceo broj n ≥ 0
// Izlaz: broj x^n
algorithm power(x, n)

   if (n == 0) then
      return 1;
   else
      return x * power(x,n-1);
```

Rekurzivni algoritmi

- ***Rekurzivni algoritam*** je algoritam koji ***poziva sam sebe*** – taj poziv može biti:
- ***Direktan*** – ukoliko se ***u njegovoj definiciji*** nalazi ***poziv samog algoritma*** koji se definiše (kao recimo *power*)
- ***Indirektan*** - ukoliko se ***u njegovoj definiciji*** nalazi ***poziv drugog algoritma*** koji pak ***sa svoje strane poziva polazni algoritam*** koji se definiše

Rekurzivni algoritmi

- ***Mehanizam pozivanja*** rekurzivnih algoritama:
- Ne razlikuje se od ***standardnog mehanizma*** za obične algoritme

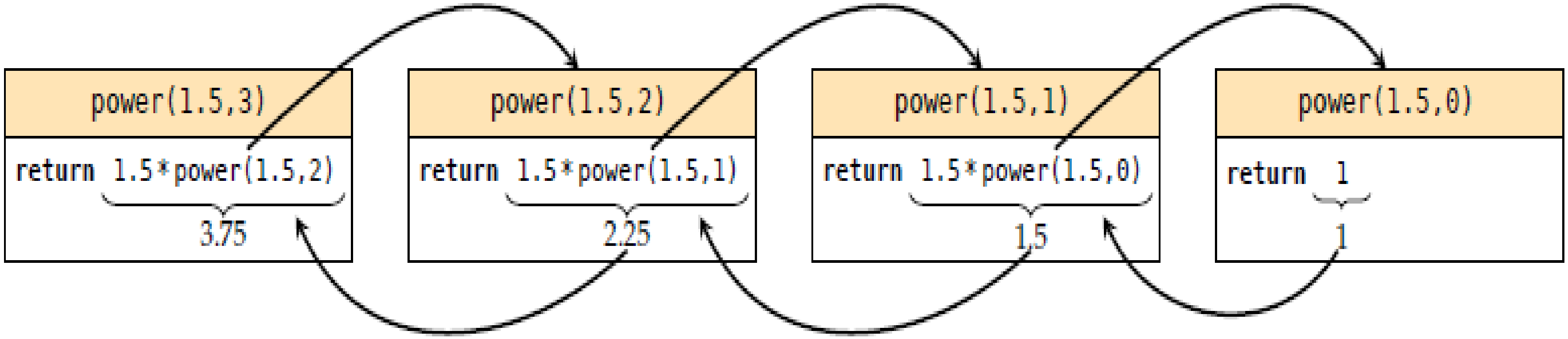
I. ***Argumenti*** u pozivu se ***dodeljuju formalnim*** ulaznim ***parametrima*** algoritma

II. ***Zatim*** se ***izvršava telo algoritma***

- ***Šta je važno*** ovde primetiti što se tiče ***pozivanja rekurzivnog algoritma?***

Rekurzivni algoritmi

- Poziv rekurzivnog algoritma ***– generiše <u>lanac poziva</u>*** istog algoritma ***sa različitim ulaznim parametrima koji definišu tekući potproblem***
- Lanac se ***<u>prekida</u>*** kad se dođe do trivijalnog potproblema – čije je rešenje očigledno
- Primer: ***power(1.5, 3)*** → ***3.75***



Rekurzivni algoritmi

- ***Rekurzivni algoritmi*** rešavaju problem njegovim ***svođenjem na sličan, prostiji, problem***
- Algoritam ***power***:
- Problem izračunavanja ***n-tog stepena*** nekog broja ***svodi na*** problem izračunavanja ***(n-1)-og stepena*** istog broja (***koji je prostiji***)
- Dalje, problem izračunavanja ***(n-1)-og stepena*** nekog broja svodi na problem izračunavanja ***(n-2)-og stepena*** istog broja (***koji je prostiji***)
- **...................**

Rekurzivni algoritmi

- ***Bez kontrole*** – ovaj postupak uprošćavanja polaznog problema bi doveo do ***beskonačnog lanca poziva*** (slično beskonačnoj petlji u iterativnim algoritmima)
- ***Bazni slučaj*** – problem u najprostijem obliku koji ***prekida lanac poziva***
- ***Rešenje*** ovog problema ***se unapred zna***
- U slučaju algoritma ***power*** – to je slučaj za ***n = 0***

Rekurzivni Euklidov algoritam

- Podsetimo se Euklidove teoreme:
- ***Teorema*** (**Euklid**). Ako su ***x*** i ***y*** pozitivni celi brojevi takvi da je ***x* ≥ *y***, onda je

$$nzd(x, y) = nzd(y, x\%y)$$

- ***Posledica***: ***svaki pozitivan ceo broj deli 0 bez ostatka***, t.j.

$$nzd(x, 0) = x$$

Rekurzivni Euklidov algoritam

- ***Iterativni algoritam (A)***:
- Primer: ***nzd(x, y), $x \geq y$ > 0*** celi brojevi
- d **= nzd(x, y)** ako

1. d je ***zajednički delilac*** za ***x*** i ***y*** (d deli oba broja x i y bez ostatka)
2. d je ***najveći*** zajednički delilac za ***x*** i **y**

- ***nzd(30, 12) = ?***

Rekurzivni Euklidov algoritam

- ***Iterativni algoritam (A)***:
- Primer: ***nzd(30, 12) = ?***

  - ***delioci za 30***: ***{1, 2, 3, 5, 6, 10, 15, 30}***
  - ***delioci za 12***: ***{1, 2, 3, 4, 6, 12}***
  - ***zajednički*** delioci za 30 i 12: ***{1, 2, 3, 6}***
  - ***najveći*** zajednički delioc - ***6***
  - ***nzd(30, 12) = 6***

Rekurzivni Euklidov algoritam

- ***Iterativni algoritam (A)***:
- ***Ideja***: počinjući ***od manjeg broja***, ***<u>proveravati sve manje brojeve</u>*** dok se ne pronađe nzd

```
// Ulaz:  pozitivni celi brojevi x i y, x ≥ y
// Izlaz: nzd(x,y)
algorithm nzd(x, y)

    d = y;
    while ((x % d != 0) || (y % d != 0)) do
        d = d - 1;

    return d;
```

Rekurzivni Euklidov algoritam

- ***Iterativni algoritam (A)***:
- ***Diskusija rešenja (A):***

- ***nzd(12378, 3054) = ?***

- ***Brojevi koji se proveravaju*** da li su delioci oba broja 12378 i 3054:

3054, 3053, 3052, 3051, … , 7, 6

- ***nzd(12378, 3054) = 6***

Rekurzivni Euklidov algoritam

- ***Iterativni algoritam (B)***:
- Zasniva se na tvrdnji Teoreme:

$$\boldsymbol{nzd(x, y) = nzd(y, x\%y)}$$

- ***nzd(12378, 3054) = ?***

Rekurzivni Euklidov algoritam

- ***Iterativni algoritam (B)***:
- ***Rešenje (B)***:

```
algorithm euclid1(x, y)

  while (y > 0) do
      z = x % y;
      x = y;
      y = z;

  return x;
```

nzd(x, y) = nzd(y, x%y)

12378, 3054

12378, 3054, 162 (***12378 % 3054*** = 162)

12378, 3054, 162, 138 (***3054 % 162*** = 138)

12378, 3054, 162, 138, 24 (***162 % 138*** = 24)

12378, 3054, 162, 138, 24, 18 (***138 % 24*** = 18)

12378, 3054, 162, 138, 24, 18, 6 (***24 % 18*** = 6)

12378, 3054, 162, 138, 24, 18, 6, 0 (***18 % 6*** = 0)

⇒ ***nzd(12378, 3054) = 6***

Rekurzivni Euklidov algoritam

- ***Iterativni algoritam***:

```
algorithm euclid1(x, y)

  while (y > 0) do
     z = x % y;
     x = y;
     y = z;

  return x;
```

x y

12378, 3054

x y

12378, 3054, 162 (***12378 % 3054*** = 162)

x y

12378, 3054, 162, 138 (***3054 % 162*** = 138)

x y

12378, 3054, 162, 138, 24 (***162 % 138*** = 24)

12378, 3054, 162, 138, 24, 18 (***138 % 24*** = 18)

12378, 3054, 162, 138, 24, 18, 6 (***24 % 18*** = 6)

x y

12378, 3054, 162, 138, 24, 18, 6, 0 (***18 % 6*** = 0)

⇒ ***nzd(12378, 3054) = 6***

Rekurzivni Euklidov algoritam

- ***Rekurzivno rešenje***:
- ***nzd(12378, 3054) = ?***

- ***Rešenje:***
- Zasniva se na ***istoj tvrdnji (Euklidova teorema)***

Rekurzivni Euklidov algoritam

- ***Rekurzivni algoritam***:
- Izračunavanje se ***podeli u <u>dva slučaja</u>***:
- ***<u>Bazni slučaj</u>***: ako je ***x%y = 0***, tada je ***nzd(x,y) = y***
- ***<u>Opšti slučaj</u>***: ako je ***x%y ≠ 0***, onda je

nzd(x,y) = nzd(y, x%y)

- ***<u>Rekurzivna definicija</u>*** NZD:

$$nzd(x,y) = \begin{cases} y, & \text{ako je } x\%y = 0 \\ nzd(y, x\%y), & \text{ako je } x\%y \neq 0 \end{cases}$$

Rekurzivni Euklidov algoritam

- ***Rekurzivno rešenje***:
- ***Rešenje:***

12378, 3054

12378, ***3054, 162*** (***12378 % 3054*** = 162)

12378, 3054, ***162, 138*** (***3054 % 162*** = 138)

12378, 3054, 162, ***138, 24*** (***162 % 138*** = 24)

12378, 3054, 162, 138, ***24, 18*** (***138 % 24*** = 18)

12378, 3054, 162, 138, 24, ***18, 6*** (***24 % 18*** = 6)

$\Rightarrow$ ***nzd(12378, 3054) = 6***

Bazni slučaj!

Rekurzivni Euklidov algoritam

- ***Rekurzivni algoritam***:
- ***Postupak nije beskonačan***, jer se ***u sledećem koraku*** uvek ***dobija broj*** (ostatak) koji je ***<u>striktno manji od prethodnog</u>*** broja (delioca)

- ***<u>Pitanje</u>:***
- Zašto ***zajednički delilac*** za ***x*** i ***y*** deli i ***x%y bez ostatka***?

Rekurzivni Euklidov algoritam

- ***Rekurzivni algoritam***:
- Zašto ***zajednički delilac*** za ***x*** i ***y*** deli i ***x%y bez ostatka***?

x predstavljen pomoću rezultata deljenja i ostatka deljenja sa ***y***:

$$x = y \cdot \lfloor x/y \rfloor + x\%y$$

d bilo koji ceo broj koji deli ***x*** i ***y*** bez ostatka:

$$\frac{x}{d} = \frac{y \cdot \lfloor x/y \rfloor}{d} + \frac{x\%y}{d}$$

Rekurzivni Euklidov algoritam

- ***Rekurzivni Euklidov algoritam za izračunavanje nzd(x, y)***:

```
// Ulaz:  pozitivni celi brojevi x i y, x ≥ y
// Izlaz: nzd(x,y)
algorithm nzd(x, y)

   if (x % y == 0) then
      return y;
   else
      return nzd(y,x % y);
```

Rekurzivni algoritam - Primer

- *Primer*: ***indeks najvećeg elementa niza***

```
// Ulaz:  niz a, njegov broj elemenata n
// Izlaz: indeks najvećeg elementa niza a
algorithm max(a, n)
```

Rekurzivni algoritam - Primer

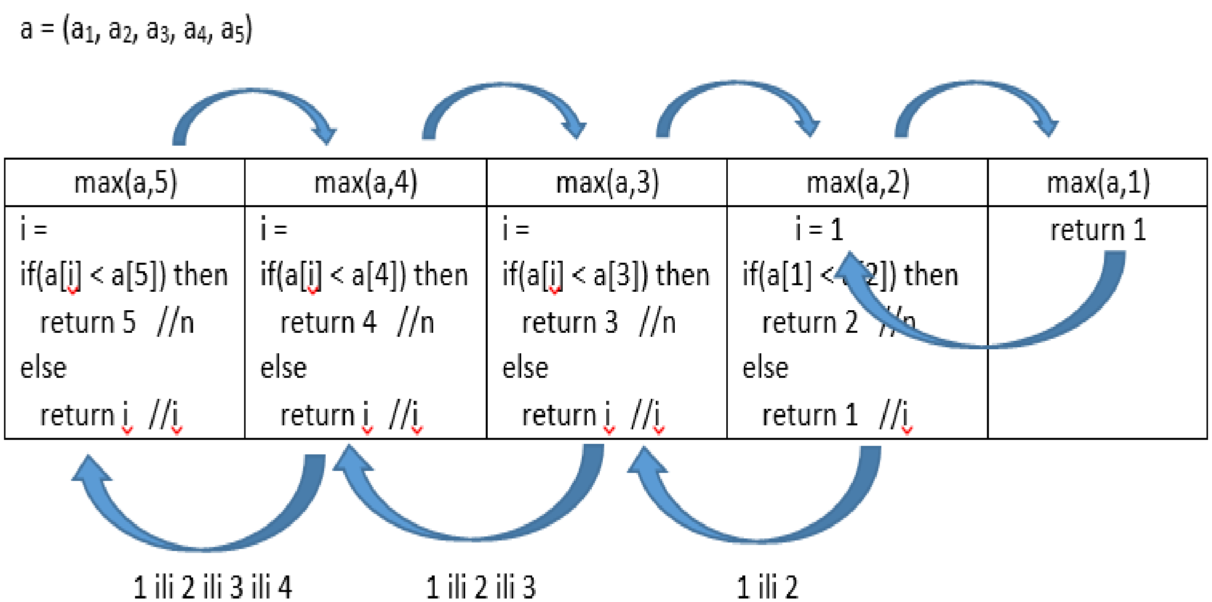
- ***Primer***: ***indeks najvećeg elementa niza***

```
// Ulaz:  niz a, njegov broj elemenata n
// Izlaz: indeks najvećeg elementa niza a
algorithm max(a, n)

    if (n == 1) then // bazni slučaj
        return 1;
    else             // opšti slučaj
        i = max(a, n-1);
        if (a[i] < a[n]) then
            return n;
        else
            return i;
```

Rekurzivni algoritmi - Primer

- Poziv rekurzivnog algoritma **– *generiše <u>lanac poziva</u>*** istog algoritma ***sa različitim ulaznim parametrima koji definišu tekući potproblem***
- Lanac se ***<u>prekida</u>*** kad se dođe do trivijalnog potproblema




```
algorithm max(a, n)

  if (n == 1) then // bazni
     return 1;
  else             // opšti
     i = max(a, n-1);
     if (a[i] < a[n]) then
        return n;
     else
        return i;
```

Rekurzivni algoritam - Primer

- ***Primer***: ***Fibonačijev niz brojeva***
- Ima brojne ***primene u*** umetnosti, ***matematici*** i računarstvu
- ***Prva dva broja*** Fibonačijevog niza brojeva su ***1 i 1***
- ***Svaki sledeći*** broj u nizu se dobija kao ***zbir prethodna dva*** broja niza
- Prema tome, početni deo Fibonačijevog niza brojeva je:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

Rekurzivni algoritam - Primer

- ***Iterativni* algoritam**:
- Algoritam koji ***izračunava n-ti broj Fibonačijevog niza***
- Da bi se izračunao ***n-ti broj*** Fibonačijevog niza

1. Može se ***početi od prva dva*** broja tog niza
2. Zatim ***primeniti iterativni postupak*** u kojem se u svakom koraku izračunava ***naredni broj u nizu*** dok se ne dobije traženi n-ti broj

Rekurzivni algoritam - Primer

- ***Iterativni* algoritam**:
- Algoritam koji ***izračunava n-ti broj Fibonačijevog niza***
- ***Ideja***: koriste se ***tri promenljive*** koje u svakom iterativnom koraku sadrže:

1. ***Prethodna dva*** broja niza (promenljive ***x i y***)
2. ***Naredni*** broj niza (promenljiva ***z***)

- ***Ulaz***: $n \geq 1$
- ***Izlaz***: ***n-ti broj*** Fibonačijevog niza (promenljiva ***z***)

Rekurzivni algoritam - Primer

- ***Iterativni algoritam***:

```
// Ulaz:  n ≥ 1
// Izlaz: n-ti broj Fibonačijevog niza
algorithm fib1(n)

   x = 1; y = 1; z = 1;
   for i = 3 to n do
      z = x + y;
      x = y;
      y = z;

   return z;
```

Rekurzivni algoritam - Primer

- ***Rekurzivni algoritam***:
- Ako brojeve Fibonačijevog niza označimo sa $f_1, f_2, f_3, \ldots\ldots\ldots$
- ***Rekurzivna definicija*** za ***n-ti*** broj Fibonačijevog niza:

$$f_1 = 1,\ f_2 = 1$$

$$f_n = f_{n-1} + f_{n-2}, \quad n > 2.$$

Rekurzivni algoritam - Primer

- ***Rekurzivni algoritam***:

```
// Ulaz:  n ≥ 1
// Izlaz: n-ti broj Fibonačijevog niza
algorithm fib2(n)

   if (n <= 2) then  // bazni slučaj
      return 1;
   else              // opšti slučaj
      return fib(n-1) + fib(n-2);
```