Dizajn i analiza algoritama

Lekcija 3

leto 2019/2020

Prof. dr Branimir M. Trenkić

Rešenje domaćeg zadatka

Konstruisati algoritam koji ***konvertuje*** ***prirodan broj*** ***u njegov*** ***oktalni zapis***. Dokazati ***korektnost*** tog algoritma.

Uputstvo:

Ulazni parametri: prirodan ***broj n*** (u dekadnom zapisu)

Izlazni parametri: ***niz*** oktalnih cifara ***b*** broja n

Napomena: ***b[1]*** je cifra na poziciji najmanje težine

Rešenje domaćeg zadatka

- Problem konverzije dekadnog prirodnog broja u oktalni zapis

- Primer ***instance problema*** konverzije dekadnog prirodnog broja u oktalni zapis:

$$n = 1463_{10}$$

- ***Rešenje*** ove ***instance***: ***b = [7, 6, 6, 2]*** (2667_8)

Rešenje domaćeg zadatka

- ***Opšti pristup***:
- ***Dizajn algoritma*** ***saglasan*** poznatom računskom ***postupku za pretvaranje*** prirodnog ***broja*** u zapise ***različitih brojnih sistema*** (binarni, oktalni, heksadecimalni)

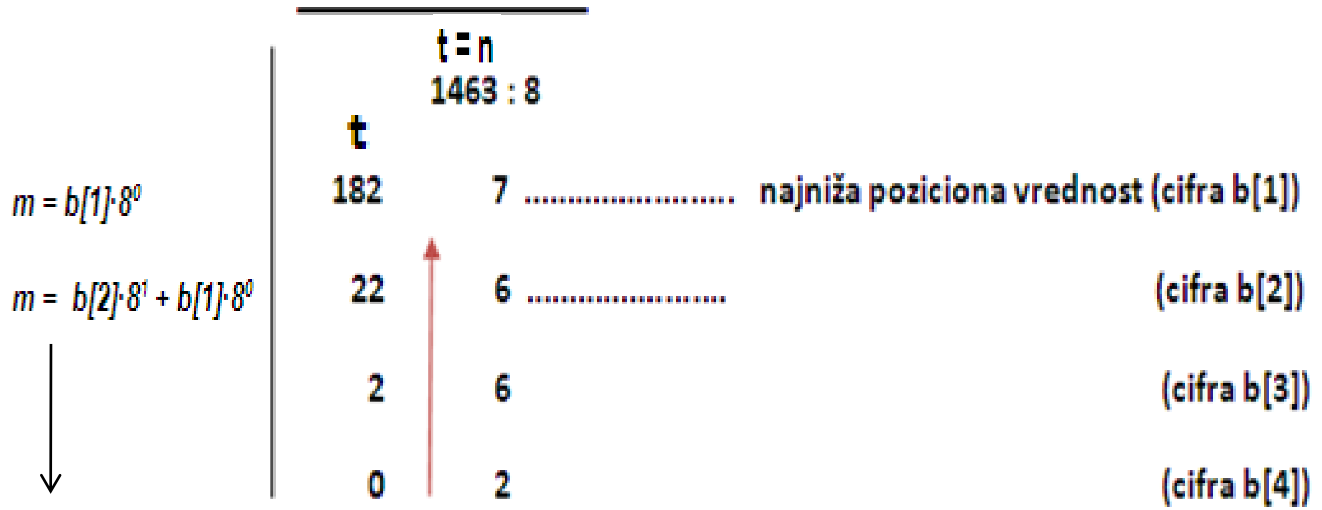
- ***Postupak*** dobijanja zapisa - ??

Rešenje domaćeg zadatka

- ***Opšti pristup***:
- ***Postupak*** dobijanja zapisa:
- ***Uzastopnim deljenjem količnika bazom*** brojnog sistema i zapisivanjem ***ostatka***
  - ***Inicijalno*** – ***količnik = n***
  - ***Količnik*** iz ***predhodne*** iteracije je ***deljenik*** u ***sledećoj***
  - ***Ostatak – cifre*** koje čine ***broj u traženom brojnom sistemu*** a koji je ekvivalent datom prirodnom broju

Rešenje domaćeg zadatka

- ***Primer za konkretni brojni sistem***:
- Konvertovati (pretvoriti) prirodni broj ***1463_{10}*** u ***oktalni ekvivalent***

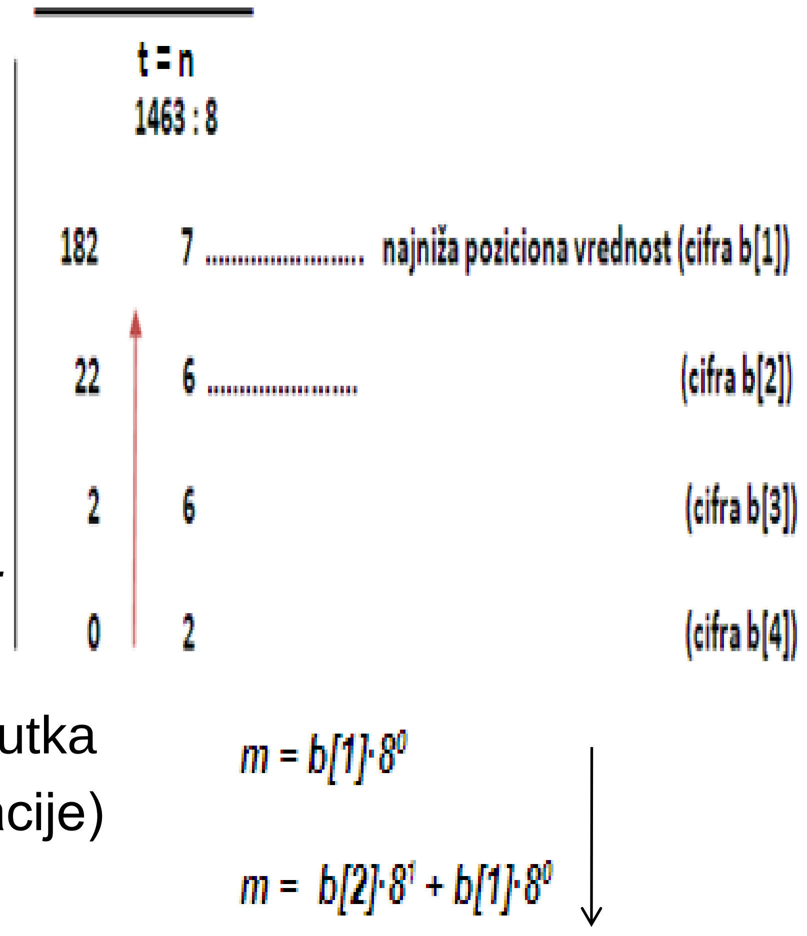


- Rešenje: n = 1463_{10} **=** 2667_8

(poslednje $m = b[4]\cdot 8^3 + b[3]\cdot 8^2 + b[2]\cdot 8^1 + b[1]\cdot 8^0$)

Rešenje domaćeg zadatka

- ***Diskusija*** algoritamskog rešenja:
- ***Ulazno-izlazni*** parametri:
  - ***n*** – prirodni broj, ***b*** - niz
- ***Pomoćne*** promenjive:
  - ***t*** – tekuća ***vrednost deljenika*** koji u ***svakom koraku*** delimo sa bazom (8)
    - Inicijalno jednak ***n***
  - ***m*** – broj čiji se ***oktalni ekvivalent*** dobija nakon ***svakog koraka***
    - Od dobijenih cifara do tog trenutka
  - ***i*** – redni broj tekućeg koraka (iteracije)



Rešenje domaćeg zadatka

- ***Diskusija*** algoritamskog rešenja:
- Iterativni algoritam
- Aritmetičke operacije:
  - Celobrojno deljenje (***/***)
  - Ostatak pri deljenju (***%***)

Rešenje domaćeg zadatka

- ***Algoritam*** u pseudo kodu:

```
// Ulaz:  prirodan broj n
// Izlaz: niz b - niz oktalnih cifara broja n
algorithm okt_cifre(n)

    t = n;           // pocetna vrednost pomocne promenljive
    i = 0;           // tekuca iteracija
    while (t > 0) do       // uslov

          i = i + 1;
          b[i] = t % 8;    // oktalna cifra dobijena
          t = t/8;         // novi kolicnik

    return b;
```

Rešenje domaćeg zadatka

- Dokaz korektnosti algoritma

1. Izlazni uslov

- Da li algoritam terminira rad, tj. da li će u nekom trenutku biti zadovoljen ***izlazni uslov while petlje***, tj. $\boldsymbol{t = 0}$?
- Niz $\boldsymbol{t_{i+1} = t_i / 8}$ je ***monotono opadajući niz*** prirodnih brojeva, te je po algebarskom principu minimalnog elementa ***ograničen odozdo nulom***

Rešenje domaćeg zadatka

2. Invarijanta petlje

- Kako doći do nje (– generalno)?
- Imamo neke ***promenljive čije se vrednosti menjaju*** u toku izvršavanja petlje
  - Tekuće vrednosti u ***pomoćne promenljive***
- Identifikovati ***odnos***
  - Njihov ***međusobni odnos***
  - Njihov ***odnos prema*** nekim ***vrednostima*** koje se ***ne menjaju*** u toku izvršavanja algoritma
- Taj odnos definisati kao uslov čija se tačnost ne menja - ***invarijanta petlje***

Rešenje domaćeg zadatka

- Dokaz <u>korektnosti algoritma</u>
- U našem primeru može se uočiti sledeći odnos koji definišemo kao ***invarijanta petlje***:

$$t \cdot 8^i + m = n$$

t:

1463 : 8

182	7	....i=1......$m = b[1]\cdot 8^0$................	$t\cdot 8^i + m = 182\cdot 8^1 + 7\cdot 8^0 = 1463 = n$
22	6	.....i=2.....$m = b[2]\cdot 8^1+b[1]\cdot 8^0$....	$t\cdot 8^i + m = 22\cdot 8^2 + 6\cdot 8^1 + 7\cdot 8^0 = 1463 = n$
2	6	.....i=3.....$m = b[3]\cdot 8^2+b[2]\cdot 8^1+b[1]\cdot 8^0$....	....
0	2	.....i=4.....$m = b[4]\cdot 8^3+b[3]\cdot 8^2+b[2]\cdot 8^1+b[1]\cdot 8^0$.....	....

Rešenje domaćeg zadatka

- Dokaz korektnosti algoritma
- ***Baza indukcije***: Provera da li je invarijanta tačna ***pre prve iteracije***
- ***Pomoćne promenljive*** imaju sledeće vrednosti: $\boldsymbol{i = 0}$, $\boldsymbol{t = n}$ i $\boldsymbol{m = 0}$ (ne postoji ni jedna cifra)
- Provera tačnosti uslova invarijante:

$$n \cdot 8^0 + 0 = n$$

$$n = n$$

Rešenje domaćeg zadatka

- Dokaz korektnosti algoritma
- ***Induktivna hipoteza***: Predpostavimo da je invarijanta petlje ***tačna*** nakon iteracije i
- ***Pomoćne promenljive*** u tom trenutku imaju vrednosti: $i \neq 0$, t i m (***b[i]*** do ***b[1]***)(to je slučaj ***pre i + 1 iteracije***)
- Znači, važi tvrdnja

$$t \cdot 8^i + m = n$$

Rešenje domaćeg zadatka

- Dokaz korektnosti algoritma
- U narednom prolasku kroz petlju (***nakon i + 1 iteracije***) dobijaju se nove vrednosti
- (novo ***i***) ***i + 1***
- (novu cifru) $b[i + 1] = t \% 8$
- (novo ***t***) $t' = t / 8$
- (novo ***m***) $m' = b[i + 1]\cdot 8^i + m = (t \% 8)\cdot 8^i + m$
- Sada treba ***za nove vrednosti proveriti da li invarijanta petlje važi***

Rešenje domaćeg zadatka

- Dokaz korektnosti algoritma
- Dokaz da za nove vrednosti pomoćnih promenljivih - invarijanta petlje i dalje važi
- Treba pokazati:

$$t' \cdot 8^{i+1} + m' = n$$

$$t' \cdot 8^{i+1} + m' = (t\ /\ 8) \cdot 8^{i+1} + (t\ \%\ 8) \cdot 8^{i} + m =$$

$$= 8^{i}(8 \cdot (t\ /\ 8) + (t\ \%\ 8)) + m = t \cdot 8^{i} + m = n$$

- Dakle, ***dokazali smo*** da izraz $t \cdot 8^i + m = n$ ne menja tačnost prolaskom kroz petlju – što znači da je to invarijanta petlje

Problem maksimalne sume podniza

- ***Često se pojavljuje*** u računarskoj biologiji prilikom ***analize DNK*** ili ***proteinske sekvence***
- Interesantan problem zato ***postoji veliki broj algoritama*** koji ga rešavaju i različite su efikasnosti
- ***Najsporiji – <u>kvadratni</u> algoritam*** za rešenje na bazi definicije problema
- Šta smatramo ***podnizom*** nekog niza?
- ***<u>Neprekidni podniz</u>*** nekog niza ***a*** – niz koji se sastoji od nula ili više ***<u>uzastopnih elemenata</u>*** niza ***a***

Problem maksimalne sume podniza

Na primer,

a = [2, -1, 1, 3, -4, -6, 7, -2, 3, -5]

- ***Korektni*** neprekidni podnizovi:

  [1, 3, -4, -6, 7] i ***[-1]***

- ***Nekorektni*** neprekidni podnizovi:

  [2, 1, 3]

- U rešavanju problema ***<u>podrazumevamo</u>*** da su ***podnizovi – <u>neprekidni podnizovi</u>***

Problem maksimalne sume podniza

- ***Oznake***:
- Dati podniz ***a_i, ..., a_j*** niza ***a***. ***Sumu*** tog ***podniza*** označavamo ***$S_{i,j}$*** i predstavlja ***običan zbir*** svih elemenata podniza:

$$S_{i,j} = a_i + a_{i+1} + \cdots + a_j = \sum_{k=i}^{j} a_k$$

- Napomena: ***<u>prazan niz</u>*** je neprekidni niz svakog niza – po definiciji: ***suma*** praznog niza je ***<u>jednaka nuli</u>***

Problem maksimalne sume podniza

- ***Definišimo problem***:

Problem maksimalne sume (neprekidnog) podniza sastoji se u tome da se ***<u>odredi najveća suma</u>*** *svih neprekidnih podnizova datog niza*

- ***<u>Ulazni parametri</u>***: niz celih brojeva ***a*** i broj ***n*** (dužina niza)
- ***<u>Izlazni parametri</u>***: ceo broj ***M*** (pozitivan ili 0)

Problem maksimalne sume podniza

- Primer ***instance problema*** maksimalne sume podniza:

a = [2, -1, 1, 3, -4, -6, 7, -2, 3, -5]

- ***Rešenje*** ove ***instance***: ***8*** (zbir elemenata podniza ***[7, -2, 3]***)
- ***Napomena***: ***elementi*** datog niza mogu biti ***pozitivni***, ***negativni*** ili ***jednaki nuli***
- Šta je rešenje ako su (***I***) svi elementi pozitivni ili (***II***) svi elementi negativni?

Problem maksimalne sume podniza

- Neka je ***M*** suma podniza datog niza koji ima najveću sumu – ***rešenje problema***
- ***Po definiciji*** problema:

$$M = \max_{1 \le i \le j \le n} S_{i,j}$$

- ***Napomena***: Ovaj izraz ***isključuje slučaj*** da su ***svi elementi niza negativni*** – u tom slučaju ***rešenje je 0*** što je zbir elemenata praznog podniza

Problem maksimalne sume podniza

- ***Diskusija*** algoritamskog rešenja:
- Da bi ***izračunali M*** po definiciji – treba da izračunamo sume ***svih podnizova datog niza***
- Svi <u>podnizovi</u> koji ***počinju*** u ***poziciji 1*** datog niza
- Svi <u>podnizovi</u> koji ***počinju*** u ***poziciji 2*** datog niza

  ....
- Svi <u>podnizovi</u> koji ***počinju*** u ***poziciji n*** datog niza
- Nakon toga, treba odrediti maksimalnu vrednost svih ovih suma

Problem maksimalne sume podniza

- ***Diskusija*** algoritamskog rešenja:
- Koliko je ***broj takvih podnizova*** (suma)?
  - Podnizovi koji ***počinju u poziciji 1*** – ukupan broj ***n***
  - Podnizovi koji ***počinju u poziciji 2*** – ukupan broj ***n -1***

    ....
  - Podnizovi koji ***počinju u poziciji n*** – ukupan broj ***1***
- Ukupan broj podnizova:

$$n + (n-1) + \dots\dots + 1 = n(n+1)/2$$

Problem maksimalne sume podniza

- ***Diskusija*** algoritamskog rešenja:

- Algoritamske korake možemo reorganizovati bez uticaja na konačni rezultat

1. Računamo ***parcijalne maksimalne sume***
   – Za svako ***i = 1,2,....,n*** ***u i-tom koraku izračunavamo maksimalnu parcijalnu sumu svih podnizova koji počinju u poziciji i*** (**=** b_i)
2. Tražimo maksimum od svih dobijenih parcijalnih maksimalnih suma

Problem maksimalne sume podniza

- ***Diskusija*** algoritamskog rešenja:
- Ovo rešenje podrazumeva da se ***najpre*** izračuna ***niz b*** od ***n*** elemenata

$$b_i = \max\left\{S_{i,i}, S_{i,i+1}, \ldots S_{i,n}\right\}$$

- ***Zatim*** se računa ***M*** na sledeći način

$$M = \max\left\{b_1, b_2, \ldots, b_n\right\} = \max_{i=1,2,\ldots,n} b_i$$

Problem maksimalne sume podniza

- ***Algoritam*** u pseudo kodu:

- ***Realizacija ovog postupka*** na pseudo jeziku sastoji se od ***dva algoritma***

1. Pomoćni algoritam (***B***) koji ***računa*** elemente b_i (***parcijalne maksimalne sume***)
2. ***Glavni algoritam*** (koristi prvi algoritam) ***nalazi maksimalnu vrednost*** između svih parcijalnih maksimalnih suma
   - Napomena: Glavni algoritam koristi i pomoćni algoritam ***max*** (vidi predhodni domaći zadatak!)

Problem maksimalne sume podniza

- ***Algoritam*** u pseudo kodu:
- ***Pomoćni algoritam B*** – ***računa*** elemente niza ***b***, tj. ***maksimalne parcijalne sume***
- **Ulazni parametri**:
  - ***i*** - indeks elementa niza b koji računamo
    - ujedno je to i ***početna pozicija podnizova***
  - ***a*** – niz iz koga izdvajamo podnizove
  - ***n*** – dužina niza a
- ***Izlazni parametar***:
  - b_i – i-ta maksimalna parcijalna suma

Problem maksimalne sume podniza

- ***Algoritam*** u pseudo kodu:
- ***Pomoćni algoritam B*** – računa elemente niza ***b***, tj. maksimalne parcijalne sume – koristimo ***svojstvo***:

$S_{i,j} = S_{i,j-1} + a[j]$

```
// Ulaz:  indeks i, niz a, broj elemenata n niza a
// Izlaz: element b_i niza b
algorithm B(i, a, n)

   m = 0;      // maksimalna suma podniza od pozicije i
   s = 0;      // sume S_{i,j} za j = i,...,n

   for j = i to n do   // izračunati S_{i,j} za j = i,...,n
      s = s + a[j];    // S_{i,j} = S_{i,j-1} + a_j
      if (m < s) then  // nađena veća suma
         m = s;        // ažurirati maksimalnu sumu

   return m;  // vratiti b_i
```

Problem maksimalne sume podniza

- ***Algoritam*** u pseudo kodu:
- ***Glavni algoritam***– računa maksimalni element niza b, tj. najveću maksimalnu parcijalnu sumu

```
// Ulaz:  niz a, broj elemenata n niza a
// Izlaz: maksimalna suma podniza M
algorithm mcss(a, n)

   for i = 1 to n do
      b[i] = B(i, a, n);
   M = max(b, n);

   return M;
```

Analiza algoritama

- ***Vreme izvršavanja pomoćnog algoritma B***
- Zavisi od:
- ***n*** (ukupnog broja elemenata niza)
- ***i*** početne pozicije podnizova

T(i,n) = ?

```
// Ulaz:  indeks i, niz a, broj elemenata n niza a
// Izlaz: element b_i niza b
algorithm B(i, a, n)

   m = 0;      // maksimalna suma podniza od pozicije i
   s = 0;      // sume S_{i,j} za j=i,...,n

   for j = i to n do   // izračunati S_{i,j} za j=i,...,n
      s = s + a[j];    // S_{i,j} = S_{i,j-1} + a_j
      if (m < s) then  // nađena veća suma
         m = s;        // ažurirati maksimalnu sumu

   return m;  // vratiti b_i
```

Analiza algoritama

- ***Vreme izvršavanja pomoćnog algoritma B***
- Zavisi od:
- ***n*** (ukupnog broja elemenata niza)
- ***i*** početne pozicije podnizova

```
algorithm B(i, a, n)

  m = 0;       // maks
  s = 0;       // sume

  for j = i to n do
     s = s + a[j];
     if (m < s) then
        m = s;

  return m;   // vrat
```

$$
\begin{aligned}
T(n,i) &= 1+1+(n-i+1)(1+2)+1 \\
&= 3+3(n-i+1) \\
&= 3(n-i+2).
\end{aligned}
$$

Analiza algoritama

- ***Vreme izvršavanja pomoćnog algoritma max***

```
// Ulaz: niz a, broj elemenata n niza a
// Izlaz: vrednost najveceg elementa niza a
algorithm max(a, n)

 m = a[1];      // najveci element nadjen do sada
 j = 1;         // indeks najveceg elementa

 i = 2;
 while (i <= n) do
        if (m < a[i]) then     // nadjen veci element od
                               // privremeno najveceg
            m = a[i];             // zapamti veci broj
            j = i;                // i njegov indeks
        i = i + 1;          // predji na sledeci element

 return a[j];        // vrati vrednost najveceg elementa
```

Analiza algoritama

- ***Vreme izvršavanja pomoćnog algoritma max***

```
// Ulaz: niz a, broj elemenata n niza a
// Izlaz: vrednost najveceg elementa niza a
algorithm max(a, n)

 m = a[1];      // najveci element nadjen do sada
 j = 1;         // indeks najveceg elementa

 i = 2;
 while (i <= n) do
        if (m < a[i]) then      // nadjen veci element od
                                // privremeno najveceg
             m = a[i];              // zapamti veci broj
             j = i;                 // i njegov indeks
        i = i + 1;          // predji na sledeci element

 return a[j];        // vrati vrednost najveceg elementa
```

T(n) = 1 + 1 + 1 + (n -1)(3 + 1) +1 = 4 + 4(n -1) = 4n.

Analiza algoritama

- ***Vreme izvršavanja algoritma mcss***
- Znamo vreme izvršavanja algoritama ***B*** i ***max***

```
algorithm mcss(a, n)

   for i = 1 to n do
      b[i] = B(i, a, n);
   M = max(b, n);

   return M;
```

Analiza algoritama

- ***Vreme izvršavanja algoritma mcss***
- Znamo vreme izvršavanja algoritama ***B*** i ***max***

$$T(n) = \sum_{i=1}^{n} \left(3\left(n - i + 2\right) + 1\right) + \left(4n + 1\right) + 1$$

$$= 3\sum_{i=1}^{n} \left(n - i\right) + 7n + 4n + 2$$

$$= 3\frac{\left(n-1\right)n}{2} + 11n + 2$$

$$= 1.5n^2 + 9.5n + 2.$$

```
algorithm mcss(a, n)

  for i = 1 to n do
     b[i] = B(i, a, n);
  M = max(b, n);

  return M;
```

Dakle, dobijeni algoritam pripada kategoriji kvadratnih algoritama

Problem maksimalne sume podniza

- ***Poboljšanje*** ovog algoritma
- Željeni ***maksimum*** ***M*** elemenata b_i izračunavamo ***"u hodu"***
- Umesto da prvo nađemo sve elemente skupa b – koristimo ***tehniku privremenog maksimuma***
- Ne treba nam poseban niz b – ***ušteda memorijskog prostora***
- Ovaj postupak možemo dalje poboljšati ukoliko određujemo privremeni maksimum od $S_{i,j}$ a ne od b_i.

Problem maksimalne sume podniza

- ***Poboljšanje*** ovog algoritma – pseudo kod:

```
// Ulaz:  niz a, broj elemenata n niza a
// Izlaz: maksimalna suma podniza M
algorithm mcss(a, n)

   M = 0;    // maksimalna suma podniza
   for i = 1 to n do    // izračunati b_i za i = i,...,n
      s = 0;
      for j = i to n do    // izračunati S_{i,j} za j = i,...,n
         s = s + a[j];     // S_{i,j} = S_{i,j-1} + a_j
         if (M < s) then
            M = s;

   return M;
```

Domaći zadatak

Pokazati da ova ***poboljšana varijanta*** nije ništa brža od prvobitne – naime i poboljšana verzija pripada kategoriji ***kvadratnih algoritama***.