**Љиљана Чабаркапа**

# АЛГОРИТМИ СОРТИРАЊА

Сортирање представља поступак уређења објеката неког скупа у одређеном поретку. На пример, ако је дат низ бројева $a_1$, $a_2$, ... , $a_n$, сортирање је његово уређење у неопадајућем:

$$a_1 <= a_2 <= \cdots <= a_n$$

или нерастућем поретку:

$$a_1 >= a_2 >= \cdots >= a_n.$$

Подаци се сортирају јер је са сортираним подацима лакше радити него са подацима који су у произвољном редоследу. Са сортираним објектима се сусрећемо у телефонским именицима, речницима, библиотекама, стоваришима — скоро свуда где је потребно тражити, искључити, укључити или заменити објекте који се чувају.

На жалост, не постоји алгоритам сортирања који би био најбољи у сваком случају. Чак је тешко проценити који алгоритам сортирања је најбољи у датој ситуацији јер ефикасност алгортима може зависити од низа фактора. На пример, битно је колико објеката треба сортирати, да ли се сви могу сместити у оперативну меморију или у којој мери су објекти већ сортирани.

У настави знају да буду „камен спотицања“ за већину ученика нарочито ако се њиховом овладавању не приступи на поступан начин. Најгора метода, која се не ретко користи, је да се пред ученике просто „тресне“ неки алгоритам сортирања а затим анализира. Много је ефикасније и за ђака корисније ако му се низом припремних задатака помогне да сам осети идеју и напише одговарајући алгоритам сортирања. Ученике треба уводити у ову област алгоритмима сортирања који су очигледнији, а ефикасније методе оставити за додатну наставу. По једноставности се издвајају методе Bublle sort (или „метод мехурића“) и Selection sort (метода селекције).

Нека је задатак који треба решити следећи: низ реалних бројева $a_1$, $a_2$, ... , $a_n$ уредити у неопадајући поредак, тј. $a_1 <= a_2 <= \cdots <= a_n$.

Неизбежан задатак који се сусреће у свим алгоритмима сортирања је размена вредности две променљиве. Ако ученицима већ није познат, у уводном часу треба га обрадити.

Пример 1. Написати процедуру којом се реализује размена вредности две променљиве реалног типа.

*Решење.*

```
procedure Razmeni(var A,B:real);
  var
    P:real;
  begin
    P:=A; {*sklanja vrednost promenljive A u P*}
    A:=B; {*A dobija novu vrednost, tj. B*}
    B:=P {*B dobija vrednost promenljive P, odnosno polaznu*}
          {*vrednost od A*}
  end;
```

Пример 2. Где ће се наћи највећи елемент низа бројева: 11, 2, 12, 8, 9, 3 ако идући кроз низ с лева на десно размењујемо вредности свака два суседна елемента који нису у поретку $\leqslant$?

*Одговор.* Примењујући дати поступак над елементима низа, реализују се следећа премештања:

(1) 11 размени место са 2 $(2, 11, 12, 8, 9, 3)$
(2) 12 размени место са 8 $(2, 11, 8, 12, 9, 3)$
(3) 12 размени место са 9 $(2, 11, 8, 9, 12, 3)$
(4) 12 размени место са 3 $(2, 11, 8, 9, 3, 12)$

Очигледно је да је највећи елемент низа 12 овим поступком „претицао“ све елементе који су десно од њега и стигао на крај низа.

Пример 3. Ако се над низом 2, 11, 8, 9, 3, 12 – који је резултат премештања задатих поступком из претходног примера – понови поступак из претходног примера, где ће се преместити друга по величини вредност у низу?

*Одговор.* Сада се реализују следећа премештања:

(1) 11 размени место са 8 $(2, 8, 11, 9, 3, 12)$
(2) 11 размени место са 9 $(2, 8, 9, 11, 3, 12)$
(3) 11 размени место са 3 $(2, 8, 9, 3, 11, 12)$

тако да друга по величини вредност у низу (11) долази на претпоследње место.

Ако би се наставило са наведеним поступком премештања, добио би се низ сортиран у неопадајућем поретку: 2, 3, 8, 9, 11, 12.

Пример 4. Написати програмски сегмент којим се, под претпоставком да је дата декларација низа од $N$ елемената:

```
const
  Max=100;
type
  Niz=array[1..Max] of real;
var
  I,J,N:integer;
  A:Niz;
```

реализује поступак премештања из другог примера – где идући с лева на десно размењују вредности свака два суседна елемента која нису у поретку $\leqslant$.

*Решење.*

```
for J:=1 to N-1 do
  if A[J]>A[J+1]        {*ako nije zadovoljen trazeni poredak*}
    then Razmeni(A[J],A[J+1]);  {*razmeniti vrednosti susednih*}
                                {*elemenata*}
```

Пример 5. Шта се добија понављањем програмског сегмента из претходног примера $N-1$ пута?

*Решење.* Добија се алгоритам сортирања у неопадајућем поретку

```
for I:=1 to N-1 do
  for J:=1 to N-1 do
    if A[J]>A[J+1]         {*ako nije zadovoljen trazeni poredak*}
      then Razmeni(A[J],A[J+1]);  {*razmeniti vrednosti susednih*}
                                  {*elemenata*}
```

познат под именом Bublle sort.

Ово је најједноставнија, али и најнеефикаснија варијанта ове методе, јер:

(1) сваки од $N-1$ пролаза кроз низ се непотребно одвија до последњег елемента низа – с обзиром да је после првог проласка кроз низ највећи елемент дошао на крај, у другом пролазу треба ићи до претпоследњег итд;

(2) не проверава се да ли је низ можда постао сортиран, па је сувишно наставити наведени поступак.

Пример 6. Модификовати решење из примера 5. тако да се отклони претходни недостатак означен са (1).

*Решење.* Треба обезбедити да у првом проласку кроз низ последње поређење буде $a_{n-1}$ и $a_n$, у другом $a_{n-2}$ и $a_{n-1}$, ..., $(N-1)$-ом проласку $a_1$ и $a_2$. Да би се ово постигло треба само у циклусу по $J$ за горњу границу индекса узети $N-I$.

```
for I:=1 to N-1 do
  for J:=1 to N-I do
    if A[J]>A[J+1]         {*ako nije zadovoljen trazeni poredak*}
      then Razmeni(A[J],A[J+1]);  {*razmeniti vrednosti susednih*}
                                  {*elemenata*}
```

Пример 7. Написати процедуру за сортирање којом се отклања недостатак означен са (2).

*Решење.* Увођењем логичког индикатора `Ok` кога постављамо на `true` пре сваког проласка кроз низ циклусом `for J:=1 to N-1 do ...`, ако је код свих парова суседних елемената задовољен тражени поредак – дакле низ је уређен, `Ok` задржава вредност `true` чиме се обезбеђује излазак из спољашњег `repeat` циклуса, тј. прекида сувишно понављање поступка сортирања.

```
procedure BublleSort(N:integer;var A:Niz);
  var
    I,J:integer;
    Ok:boolean;
```

```
begin
  I:=0;
  repeat
    I:=I+1;
    Ok:=true  {*indikator koji zadrzava vrednost true ako je*}
              {*svuda zadovoljen trazeni poredak:  A[J]<=A[J+1],*}
              {*tj. niz je sortiran*}
    for J:=1 to N-I do
      if A[J]>A[J+1]     {*ako nije zadovoljen trazeni poredak*}
        then
          begin
            Razmeni(A[J],A[J+1]);  {*razmeniti vrednosti*}
                                   {*susednih elemenata*}
            Ok:=false  {*Ok ukazuje da ima neuredjenih*}
                                   {*susednih elemenata*}
          end
  until Ok
end;
```

Пример 8. Написати процедуру за сортирање низа $a_1, a_2, \ldots, a_n$ у неопадајући поредак методом селекције:

– на прво место поставити најмањи у низу $a_1, a_2, \ldots, a_n$ тако што ће се прво одредити индекс најмањег елемента, а затим извршити његова размена са првим;

– на друго место поставити најмањи између $a_2, a_3, \ldots, a_n$ тако што ће се прво одредити индекс најмањег елемента, а затим извршити његова размена са другим;

– $\ldots$ ;

– на $I$-то место поставити најмањи између $a_i, a_{i+1}, \ldots, a_n$ тако што ће се прво одредити индекс најмањег елемента, а затим извршити његова размена са $I$-тим;

– $\ldots$ :

– на $(N-1)$-во место поставити мањи између $a_{n-1}$ и $a_n$ тако што ће се прво одредити индекс мањег елемента, а затим извршити његова размена са $(N-1)$-им.

*Решење.* Програмски сегмент којим се одређује индекс минималног елемента између $a_i, a_{i+1}, \ldots, a_n$ може се реализовати на следећи начин:

```
IndMin:=I;  {*pod pretpostavkom da je I-ti element minimalan,*}
            {*sacuvati mu indeks*}
for J:=I+1 to N do
  if A[J]<A[IndMin]
    then IndMin:=J;  {*korekcija indeksa minimalnog elementa,*}
                     {*jer je tekuci manji*}
```

Када је одређен индекс минималног елемента он се разменом поставља на $I$-ту позицију:

```
if IndMin<>I
  then Razmeni(A[IndMin],A[I]);  {*razmeniti vrednosti I-tog i*}
                                  {*minimalnog elementa*}
```

Понављајући претходни поступак за вредности од $I$: 1, 2, ..., $N-1$, постављају се редом вредности од најмање до највеће. Одговарајућа процедура се може написати на следећи начин:

```
procedure SelectionSort(N:integer;var A:Niz);
  var
    I,J,IndMin:integer;
  begin
  for I:=1 to N-1 do
    begin
      IndMin:=I;  {*pod pretpostavkom da je I-ti element*}
                  {*minimalan, sacuvati mu indeks*}
      for J:=I+1 to N do
        if A[J]<A[IndMin]
          then IndMin:=J;  {*korekcija indeksa minimalnog*}
                           {*elementa, jer je tekuci manji*}
      if IndMin<>I
        then Razmeni(A[IndMin],A[I]);  {*razmeniti vrednosti I-tog*}
                                       {*i minimalnog elementa*}
    end
  end;
```

## ЗАДАЦИ

1. Над низом који се састоји од шест целих бројева: 5, 4, 6, 3, 8, 9 примењено је сортирање методом Bublle sort у неопадајућем поретку. У ком ће поретку бити бројеви после две итерације (пролаза кроз низ) сортирања?
   (A) 4 5 3 6 8 9  (B) 4 6 3 5 8 9  (C) 4 3 6 5 9 8  (D) 4 3 5 6 8 9

2. Модификовати алгоритме Bublle sort и Selection sort тако да се дати низ сортира у нерастућем поретку.

3. Написати програм којим се дати низ међусобно различитих реалних бројева уређује у тестерастом поретку са првим „зубом“ окренутим на горе: $a_1 < a_2 > a_3 < a_4 > \cdots$.

4. Написати програм којим се на основу резултата $N$ тркача на 100 m датих низом $a_1$, $a_2$, ..., $a_n$ (индекс одговара стартном броју) формира низ $b_1$, $b_2$, ..., $b_n$ где је вредност $b_j$ индекс (стартни број) такмичара који се пласирао на $J$-то место.