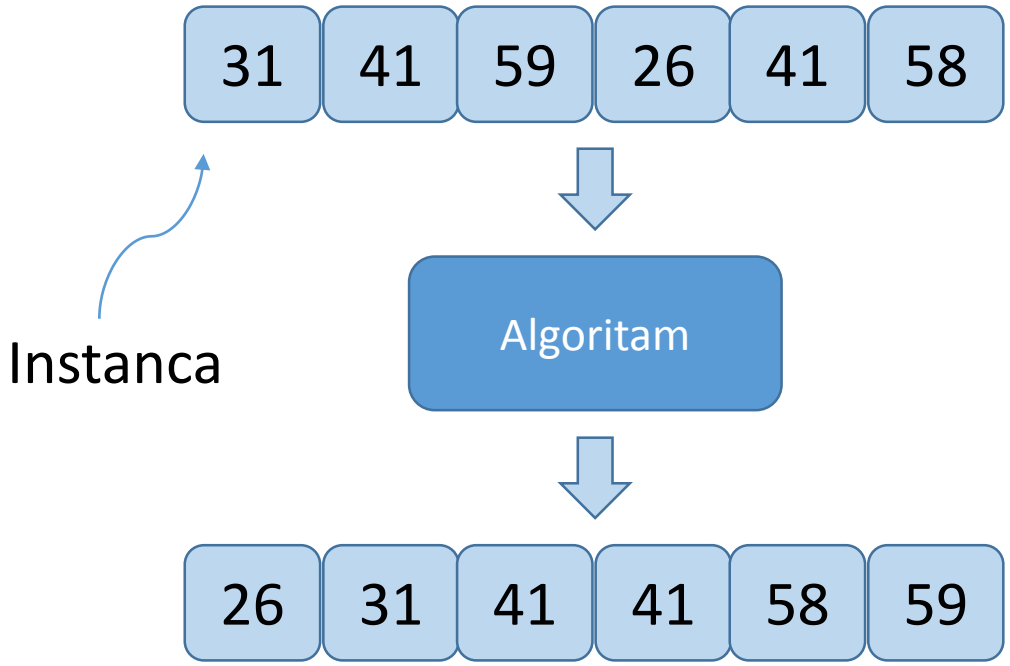
Projektovanje algoritama

L03. Dizajn i analiza algoritama. Insertion Sort. Merge Sort

Problem sortiranja



Problem sortiranja:

Ulaz: niz *N* brojeva $[a_1, a_2, ..., a_N]$

Izlaz: permutacija ulaznog niza
$[a'_1, a'_2, ..., a'_N]$

takva da je:
$a'_1 <= a'_2 <= ... <= a'_N$

Insertion Sort

- Počinjemo sa praznim nizom.
- Elemente preuzimamo jedan po jedan sa ulaza.
- Novi element ubacujemo u niz tako što
  - poredimo element sa svakim elementom u nizu, s desna na levo
  - element ubacujemo na pravo mesto tako da je niz u svakom trenutku sortiran
- Algoritam završava kada prestane biti novih elemenata na ulazu.

Insertion Sort - Pseudokod

```
INSERTION-SORT(A)
for j = 1 to A.length - 1
    key = A[j]
    i = j - 1
    while i >= 0 and A[i] > key
        A[i+1] = A[i]
        i = i - 1
    A[i+1] = key
```

http://visualgo.net/sorting

Kako dokazati tačnost algoritma?

Loop invariant (invarijanta petlje) – osobina koja se ne menja tokom izvršenja petlje.

1. **Inicijalizacija** – osobina je tačna pre prve iteracije petlje.
2. **Održavanje** – ukoliko je osobina tačna pre date iteracije petlje, ostaje tačna i pre naredne iteracije petlje.
3. **Terminacija** – kada se petlja završi, osobina nam daje korisnu posledicu koja nam pomaže da dokažemo da je algoritam tačan.

Matematička indukcija!?

Dokaz tačnosti Insertion Sort algoritma

Loop invariant:
Podniz A[0 .. j-1] sadrži sortirane elemente koji su originalno bili u istom podnizu.

1. **Inicijalizacija** j = 1
2. **Održavanje**
3. **Terminacija** A[0 .. n-1] je sortiran

Analiza algoritma Insertion Sort

Za analizu bilo kog algoritma neophodan je **model**. Mi ćemo koristiti pretpostavku da se naši algoritmi izvršavaju na **realnom računaru**.

RAM-model

Elementarne operacije (u konstantnom vremenu):

- osnovne ALU operacije (aritmetičke, logičke, pomeranja)
- load, store
- grananje (skok), poziv i povratak iz potprograma

Analiza algoritma Insertion Sort

Veličina ulaza – broj elemenata u ulaznim podacima, broj bita u ulaznom podatku, ...

Vreme izvršavanja – broj elementarnih operacija koje se izvršavaju

Analiza algoritma Insertion Sort

```
INSERTION-SORT(A)
for j = 1 to A.length - 1
    key = A[j]
    i = j - 1
    while i >= 0 and A[i] > key
        A[i+1] = A[i]
        i = i - 1
    A[i+1] = key
```

Best case

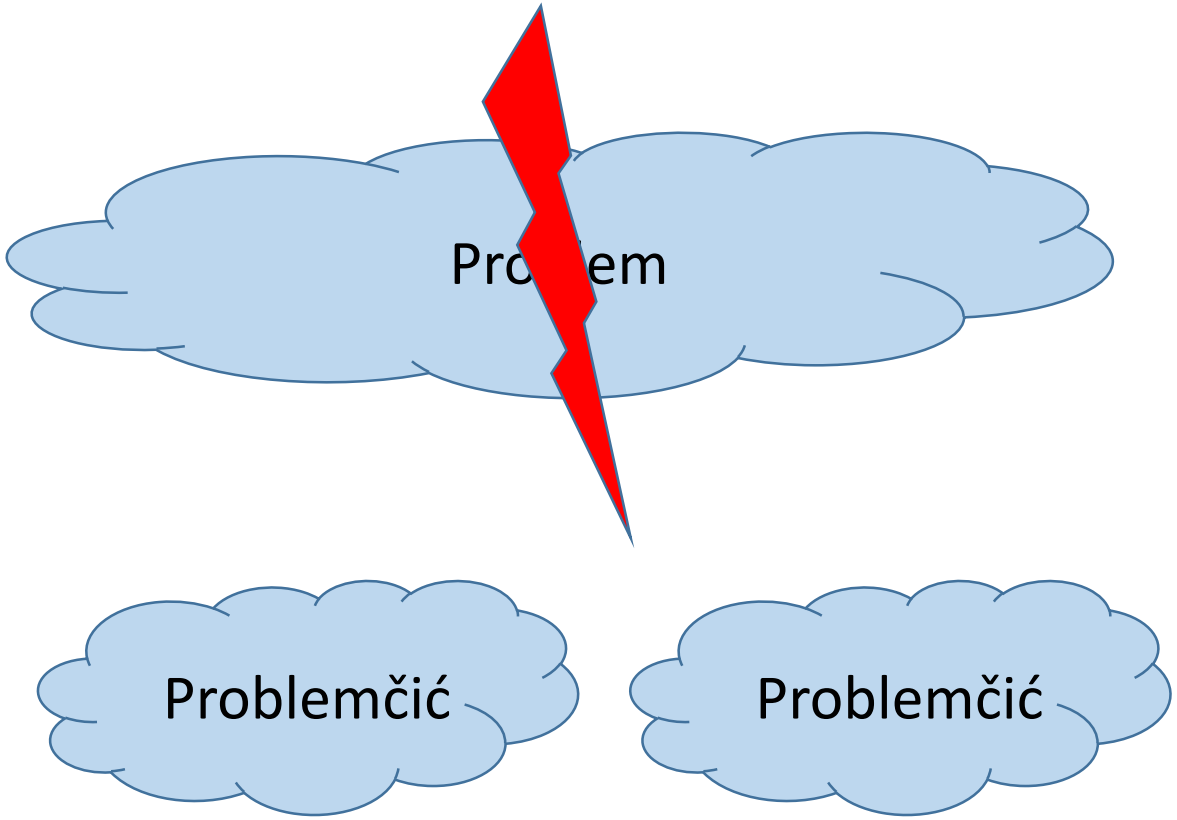
$$T(n) = an + b$$

Worst case

$$T(n) = an^2 + bn + c$$

Takmičenje

“Zavadi pa vladaj”



"Zavadi pa vladaj"

Divide – podelimo problem na nekoliko manjih verzija istog problema.

Conquer – rešimo manje verzije problema rekurzivnim pristupom.

Combine – spojimo rešenja manjih verzija problema u veliko rešenje.

Divide-and-Conquer

Merge Sort


```
MERGE (A, p, q, r)
n1 = q - p + 1
n2 = r - q
new L[1..n1+1]
new R[1..n2+1]
for i = 1 to n1
    L[i] = A[p + i - 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1+1] = Inf
R[n2+1] = Inf
i = j = 1
for k = p to r
    if L[i] <= R[j]
        A[k] = L[i]
        i = i + 1
    else
        A[k] = R[j]
        j = j + 1
```

Merge Sort

Loop invariant

Na početku petlje, A[p .. k-1] sadrži k-p najmanjih elemenata nizova L i R sortiranih u rastućem redosledu.

```
MERGE-SORT (A, p, r)
if p < r
    q = ⌊(p+r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q+1, r)
    MERGE(A, p, q, r)
```

http://visualgo.net/sorting

Analiza algoritma Merge Sort

Opšti slučaj problema koji rešavamo **divide-and-conquer** pristupom:

a potproblema veličine **n/b**

$$T(n) = \begin{cases} \theta(1), & n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) \end{cases}$$

D(n) – vreme podele problema
C(n) – vreme spajanja rešenja

Analiza algoritma Merge Sort

$$D(n) = \theta(1)$$

$$C(n) = \theta(n)$$

$$\theta(n) + \theta(1) = \theta(n)$$

$$\boldsymbol{T(n) = \theta(n \lg n)}$$

$$T(n) = \begin{cases} \theta(1), & n = 1 \\ 2T\left(\frac{n}{2}\right) + \theta(n), & n > 1 \end{cases}$$

Vežbe

1. Izmeniti INSERTION-SORT algoritam tako da sortira niz u **opadajućem** redosledu.

2. Dizajnirati algoritam **linearne pretrage elementa** i dokazati tačnost algoritma LINEAR-SEARCH.

3. Dizajnirati algoritam za **sabiranje dva binarna broja** (na binarnom nivou).

Vežbe

1. Dizajn i analiza algoritma **SELECTION-SORT**. http://visualgo.net/sorting

2. Analizirati algoritam **LINEAR-SEARCH**.

Vežbe

1. Napisati rekurentnu jednačinu za INSERTION-SORT realizovan rekurzivno.

2. Dizajnirati algoritam **binarne pretrage elementa**, napisati rekurentnu jednačinu i predvideti složenost algoritma BINARY-SEARCH.

3. Analizirati algoritam za sortiranje BUBBLE-SORT.