Algoritmi in podatkovne strukture 1

Visokošolski strokovni študij Računalništvo in informatika

Urejanje (s primerjavami)

Jurij Mihelič, UniLj, FRI

Urejanje

- Urejanje podatkov
  - **nestrukturirani** podatki
    - števila (enaka dolžina)
    - nizi (različna dolžina)
  - **strukturirani** podatki
    - zapisi (*record*, *structure, class, object*)
    - ključ in (satelitski) podatek
  - časovna zahtevnost
    - primerjava elementov
    - zamenjava elementov

Pomembnost urejanja

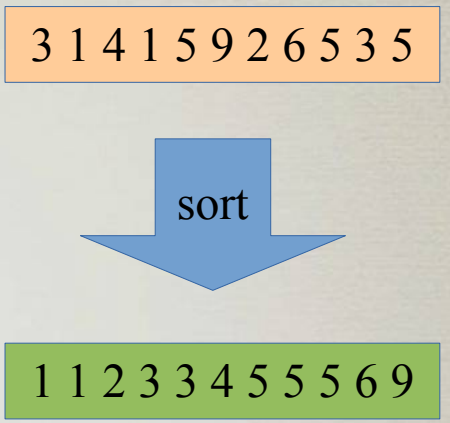
- Praktična uporabnost
  - neposredno
    - urejanje rezultata računalniške obdelave
  - posredno
    - kot podprogram v algoritmih
- Teoretična uporabnost
  - spodnja meja = zgornja meja

Pomembnost urejanja

- Inženiring algoritmov
  - ustavljanje rekurzije
  - predpomnilnik
- Predstavitev podatkov
  - urejanje s predpostavkami

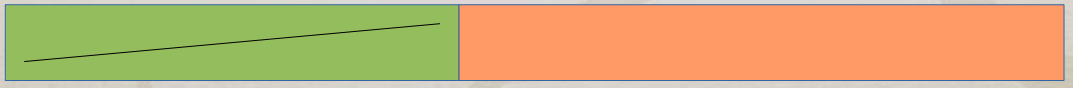
Problem urejanja

- Naloga
  - zaporedje števil $a = [a_0, a_1, \ldots, a_{n-1}]$
- Rešitev
  - permutacija $a' = [a'_0, a'_1, \ldots, a'_{n-1}]$
  - zaporedja a
  - kjer velja $a'_0 \leq a'_1 \leq \ldots \leq a'_{n-1}$
- Izvedba zaporedja
  - **polje**, povezan seznam, datoteka



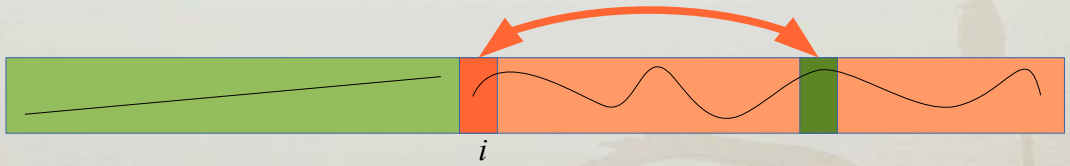
Navadna urejanja

- Navadna urejanja
  - urejanje z **izbiranjem** (*selection sort*)
  - urejanje z **vstavljanjem** (*insertion sort*)
  - urejanje z **zamenjavami** (*bubble* sort)
  - …
  - shema vseh navadnih urejanj
    - urejen seznam gradimo postopoma
    - levi del polja … urejeni del seznama
    - desni del polja … še neurejeni del seznama



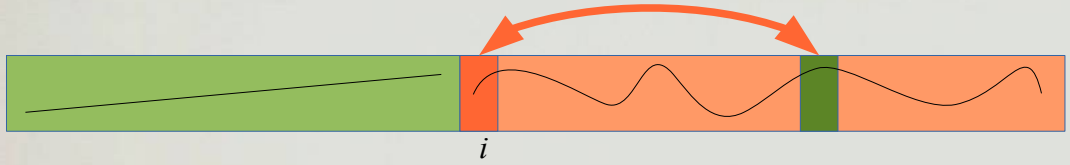
Navadno izbiranje

- Ideja algoritma
  - na vsakem koraku poiščemo najmanjši element v neurejenem delu
  - in ga *dodamo* na konec urejenega dela
  - polje: dodajanje → zamenjava
- Sled algoritma



Navadno izbiranje

- Pravilnost algoritma
  - zančna invarianta
  - vsi elementi v urejenem delu so urejeni in manjši od elementov v neurejenem delu



Navadno izbiranje

- Zahtevnost
  - št. primerjav: $n(n-1)/2 = \Theta(n^2)$
  - št. zamenjav: $n-1 = \Theta(n)$
- Izboljšave?
  - hkratno iskanje min in max
  - urejanje s kopico

Navadno izbiranje

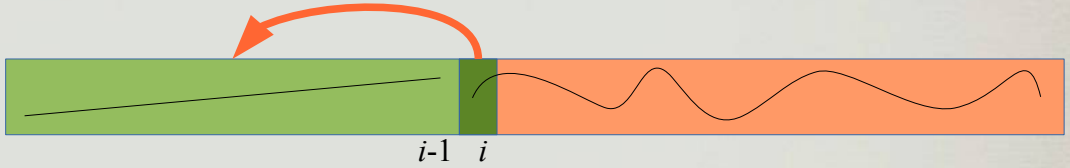
- Psevdokoda

Navadno izbiranje

```
fun selectionSort(a) is
    for i = 0 to n - 2 do
        m = i
        for j = i + 1 to n - 1 do
            if a[j] < a[m] then m = j
        swap(a, i, m)
    endfor
end
```

Navadno vstavljanje

- Ideja algoritma
  - vzamemo prvi element iz neurejenega dela in
  - ga *vstavimo* na pravo mesto v urejeni del
- Sled



Navadno vstavljanje

- Zahtevnost
  - best: $n - 1 = O(n)$
  - **worst:** $n(n-1)/2 = O(n^2)$
  - avg: $n(n-1)/4 = O(n^2)$
- Izboljšave?
  - dvojiško iskanje mesta vstavljanja
  - Shellsort

Navadno vstavljanje

- Pravilnost algoritma
  - zančna invarianta (zunanja zanka)
    - v i-ti iteraciji je tabela $a[0, 1, \ldots, i-1]$ urejena
    - torej $a_0 < a_1 < \ldots < a_{i-1}$

Navadno vstavljanje

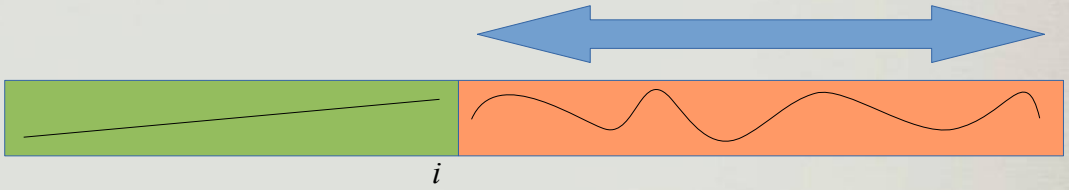
- Psevdokoda

Navadno vstavljanje

```
fun insertionSort(a) is
    for i = 1 to n-1 do
        k = a[i]
        j = i
        while j > 0 and a[j-1] > k do
            a[j] = a[j-1]
            j = j - 1
        endwhile
        a[j] = k
    endfor
end
```

Navadne zamenjave

- Ideja algoritma
  - sistematično primerjamo vsak element z vsakim
  - primerjamo paroma zaporedne elemente
- Sled
- Psevdokoda

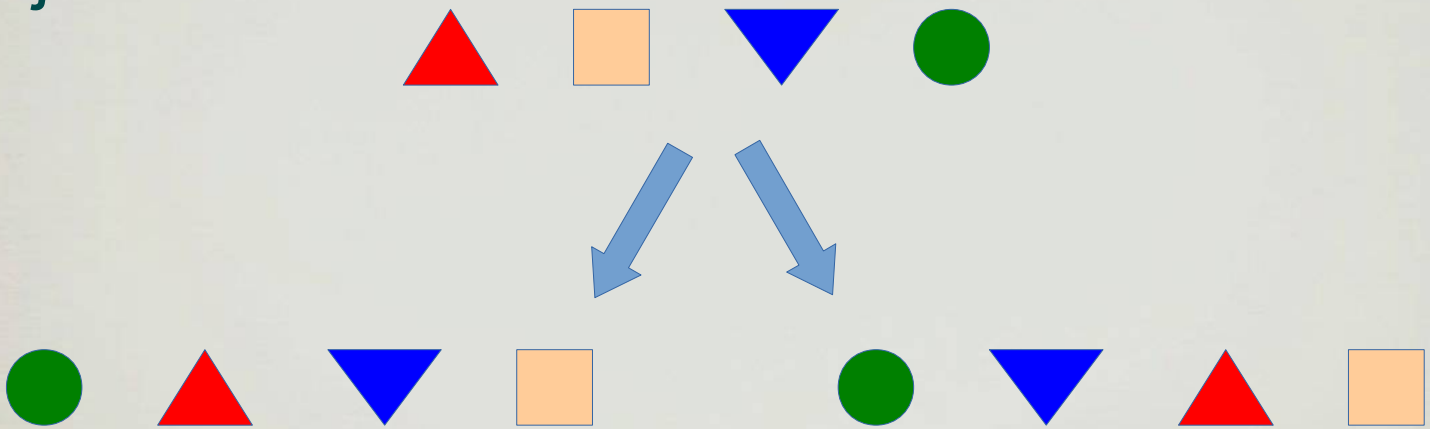


Navadne zamenjave

- Zahtevnost
  - št. primerjav: $n(n-1)/2 = \Theta(n^2)$
  - št. zamenjav: od 0 do $n(n-1)/2 = O(n^2)$
- Izboljšave?
  - detekcija urejenosti
  - izmenično urejanje

Stabilnost urejanja

- Stabilnost
  - strukturirani podatki (urejanje po ključu)
  - ohranja prvotni vrstni red pri elementih z enakim ključem



- Uporaba
  - urejanje po več ključih

Napredna urejanja

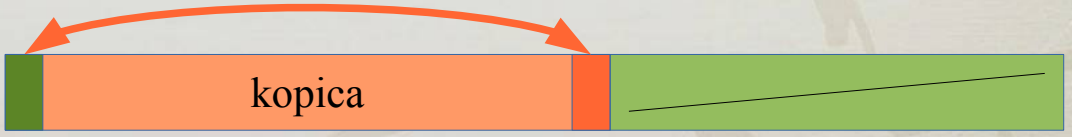
- Napredna urejanja
  - urejanje s **kopico** (*heap* sort)
  - urejanje z **zlivanjem** (*merge* sort)
  - **hitro** urejanje (*quicksort*)
  - …

Urejanje s kopico

- Ideja algoritma (*heapsort*)
  - nadgradimo urejanje z izbiranjem
  - za iskanje pravega elementa uporabimo kopico
  - 1. poskus:
    - prvi del polja je urejeni seznam
    - drugi del polja je kopica

Urejanje s kopico

- Ideja algoritma (*heapsort*)
  - nadgradimo urejanje z izbiranjem
  - za iskanje največjega elementa uporabimo kopico
  - kopico zgradimo v prvem delu tabele
  - drugi del tabele je urejeni del
  - ponavljamo
    - zamenjaj koren kopice z zadnjim elementom kopice
    - ugreznemo koren
- Sled



Urejanje s kopico

- Psevdokoda
- Zahtevnost: $O(n \lg n)$

Urejanje s kopico

```
fun heapSort(a) is
    ;; zgradi kopico
    for i = n / 2 - 1 to 0
        siftDown(a, i)

    ;; urejaj
    while last => 1 do
        swap(a, 0, last)
        last -= 1
        siftDown(0)
```

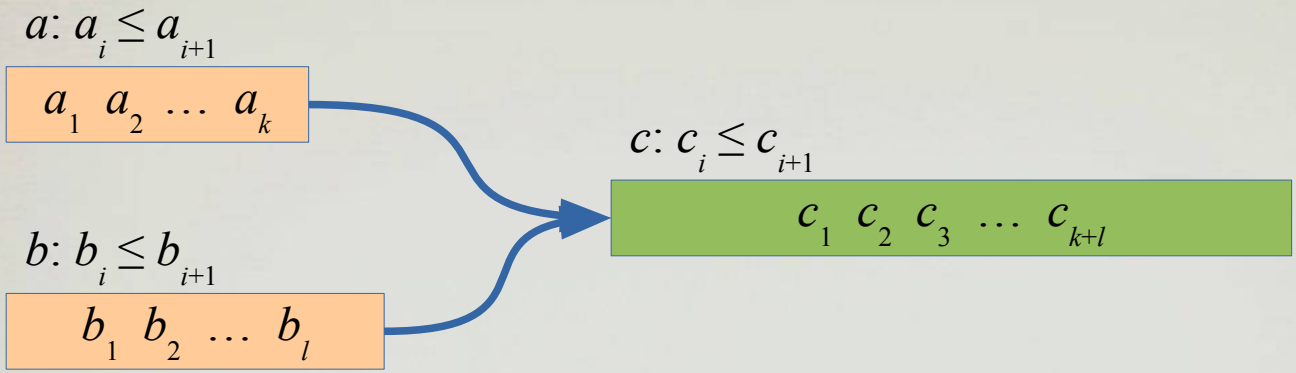
John von Neumann, 1903-1957

Urejanje z zlivanjem

- Ideja algoritma – deli & vladaj
  - tabelo razdelimo na dve polovici
  - rekurzivno uredimo obe podtabeli
  - zlijemo obe urejeni podtabeli

Urejanje z zlivanjem

- Zlivanje urejenih podtabel



- Ideja algoritma
  - hkratni zaporedni sprehod po zaporedjih
- Zahtevnost zlivanja
  - $\Theta(k+l)$

Urejanje z zlivanjem

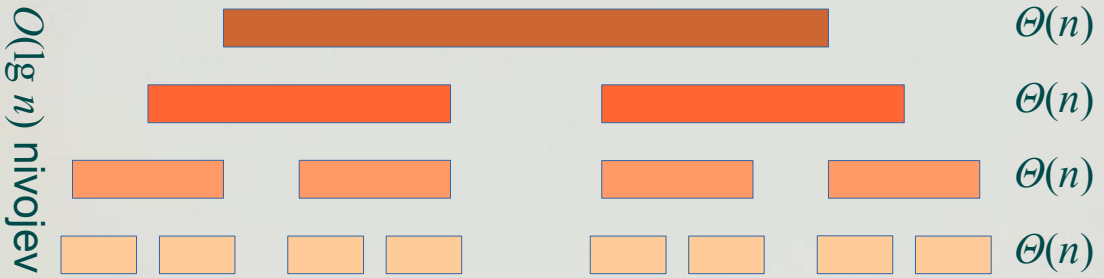
- Psevdokoda

Urejanje z zlivanjem

```
fun mergesort(a) is
    if a.length <= 1 then return a
    middle = (a.length - 1) / 2
    left = mergesort(a[0 ... middle])
    right = mergesort(a[middle+1 ... a.length-1])
    return merge(left, right)
end
```

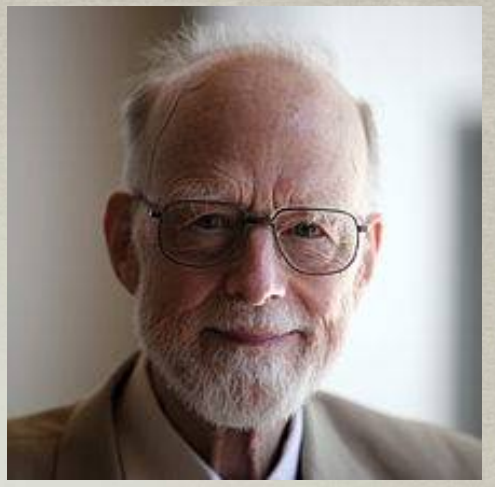
Urejanje z zlivanjem

- Zahtevnost algoritma
  - Kako globoka je lahko največ rekurzija?
  - Koliko dela je **v celoti** na vsakem nivoju rekurzije?



Hitro urejanje

C. A. R. Hoare, 1934

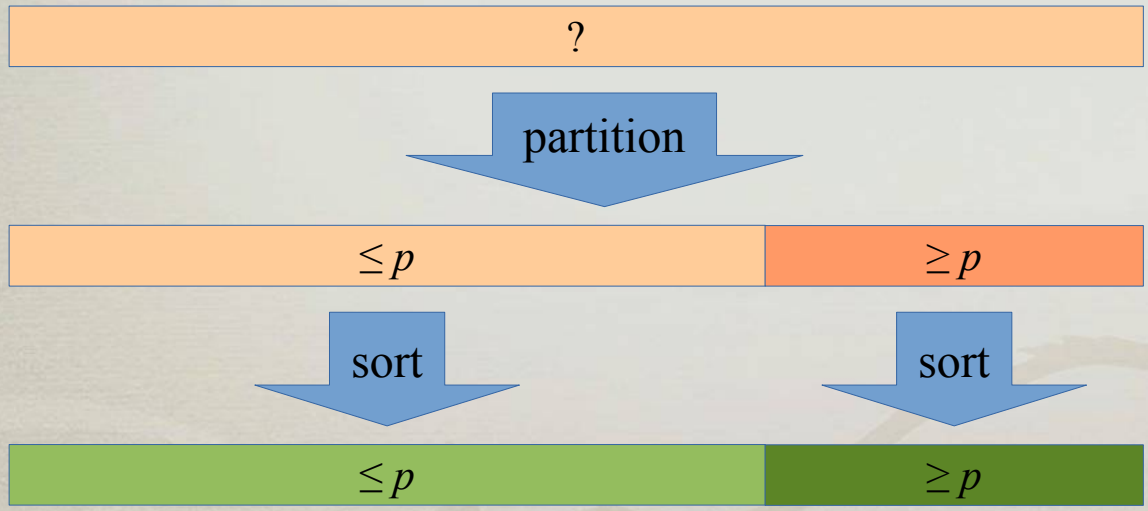


Quicksort, 1960

- Quicksort
  - eden izmed najpogosteje uporabljanih algoritmov za urejanje
  - splošen, uporablja primerjave
  - algoritem deluje neposredno v tabeli
  - potrebuje malo dodatnega prostora
  - dobro deluje za različne vrste podatkov
  - v povprečju zelo hiter
  - previdno pri implementaciji

Hitro urejanje

- Ideja algoritma – deli & vladaj
  - tabelo porazdelimo na dva dela:
    - s pomočjo poljubnega elementa p (pivot)
    - levi del vsebuje elemente $\leq p$ in desni elemente $\geq p$
  - rekurzivno uredimo obe podtabeli



Hitro urejanje

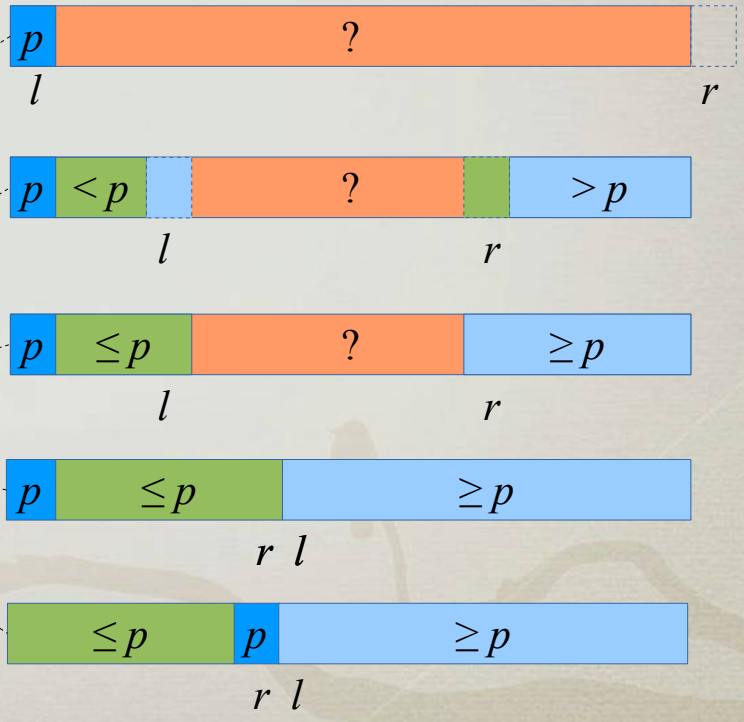
- Porazdeljevanje
  - veliko načinov
    - izbira pivota
    - z dodatnim poljem
    - križanje kazalcev
    - enozančno
    - uporaba čuvajev
    - pogoji v zankah
    - ipd.

Hitro urejanje

- Porazdeljevanje

Porazdeljevanje

```
p = a[left]
l = left; r = right + 1
while true do
    do l++ while a[l] < p and l < right
    do r-- while a[r] > p
    if l >= r then break
    swap(a, l, r)
endwhile
swap(a, left, r)
```



Hitro urejanje

Hitro urejanje

```
fun partition(a, left, right) is
    p = a[left]
    l = left; r = right + 1
    while true do
        do l++ while a[l] < p and l < right
        do r-- while a[r] > p
        if l >= r then break
        swap(a, l, r)
    endwhile
    swap(a, left, r)
    return r
end

fun quicksort(a, left, right) is
    if left >= right then return
    r = partition(a, left, right)
    quicksort(a, left, r - 1)
    quicksort(a, r + 1, right)
end
```

Hitro urejanje

- Sled
- Zahtevnost
  - best: $O(n \lg n)$
  - worst: $O(n^2)$
  - average: $O(n \lg n)$
- Prostorska zahtevnost
  - $O(n)$
  - skrbna implementacija: $O(\lg n)$
    - na sklad damo večji interval
    - repno rekurzijo spremenimo v zanko

Hitro urejanje

- Realni podatki
  - pogosto že (delno) urejena zaporedja
  - zahtevnost lahko blizu najslabše
- Izbira pivota
  - levi, desni, srednji element
  - mediana treh, petih, itd.
  - randomizacija – naključni pivot
    - `swap(a, left, random)`
    - `pivot = a[random]`
      - prekoračitev obsega, pazljivo pri izvedbi
    - pričakovana zahtevnost: $O(n \lg n)$

Hitro urejanje

- Realni podatki
  - pogosto že (delno) urejena zaporedja
  - zahtevnost lahko blizu najslabše
- Izbira pivota
  - levi, desni, srednji element
  - mediana treh, petih, itd.
  - randomizacija – naključni pivot
    - `swap(a, left, random)`
    - `pivot = a[random]`
      - prekoračitev obsega, pazljivo pri izvedbi
    - pričakovana zahtevnost: $O(n \lg n)$

Inženiring algoritmov

- Stabilnost
  - vstavljanje (IS), zamenjave (BS), zlivanje (MS)
- Velikost zaporedja
  - majhno: vstavljanje (IS)
  - veliko: veliki trije (QS, MS, HS)
- Rekurzija
  - nikoli do konca (MS, QS)
  - ustavimo z vstavljanjem (IS)

Ostalo

- State of the art
  - 2002, TimSort: MS+IS, python/java
  - 2009, Jaroslavski dvo-pivotno hitro urejanje
    - Java za primitivne tipe, ustavljanje QS z IS pri n=47
  - 2014, 5-pivotno hitro urejanje, predpomnilnik
    - Kushagra, Lopez-Ortiz, Munro, Qiao
- Zunanje urejanje
  - kadar tabela ne gre v pomnilnik
  - zlivanje: navadno/naravno, …, polifazno, kaskadno

Povzetek

Vrsta urejanja	Zahtevnost	Razno
Navadno vstavljanje	$O(n^2)$, best: $O(n)$	stabilno
Navadno izbiranje	$\Theta(n^2)$	
Navadne zamenjave	$\Theta(n^2)$	stabilno
Urejanje s kopico	$\Theta(n \log n)$	
Urejanje z zlivanjem	$\Theta(n \log n)$	stabilno, ni *in-place*, dodatni prostor
Hitro urejanje	$O(n^2)$, avg: $\Theta(n \log n)$	randomizacija, dodatni prostor
Urejanje s štetjem	$O(n + m)$	stabilno, končna množica
Korensko urejanje	$O(d(n + m))$	stabilno, končna množica
Urejanje s koši	$O(n^2)$, avg: $\Theta(n)$	stabilno?, enakomerno