

Algoritmi in podatkovne strukture 1

Visokošolski strokovni študij Računalništvo in informatika

Urejanje
(s primerjavami)



Urejanje

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

Urejanje

- Urejanje podatkov
 - primerjanje elementov
 - konstanten ali spremenljiv čas
 - zamenjava elementov
 - konstanten ali spremenljiv čas

Pomembnost urejanja

- Praktična uporabnost
 - neposredno
 - posredno
 - kot podprogram v algoritmih
- Teoretična uporabnost
 - spodnja meja = zgornja meja
- Inženiring algoritmov
 - ustavljanje rekurzije
 - predpomnilnik
- Predstavitev podatkov
 - urejanje s predpostavkami



Problem urejanja

- Naloga

- zaporedje števil $a = [a_0, a_1, \dots, a_{n-1}]$

- Rešitev

- permutacija $a' = [a'_0, a'_1, \dots, a'_{n-1}]$
- zaporedja a
- kjer velja $a'_0 \leq a'_1 \leq \dots \leq a'_{n-1}$

- Izvedba zaporedja

- **polje**, povezan seznam, datoteka

3 1 4 1 5 9 2 6 5 3 5

sort

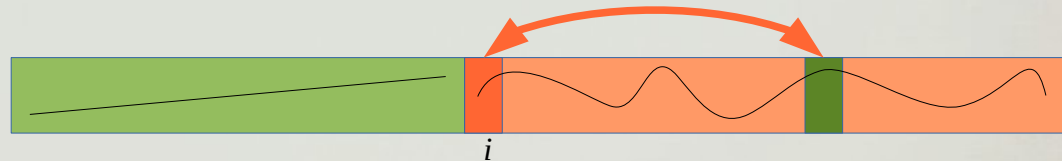
1 1 2 3 3 4 5 5 5 6 9

Algoritmi za urejanje

- Navadna urejanja
 - Urejanje z **izbiranjem** (*selection sort*)
 - Urejanje z **vstavljanjem** (*insertion sort*)
 - Urejanje z **zamenjavami** (*bubble sort*)
 - urejanje z mehurčki
 - ...
- Napredna urejanja
 - Urejanje s **kopico** (*heap sort*)
 - Urejanje z **zlivanjem** (*merge sort*)
 - **Hitro** urejanje (*quicksort*)
 - ...

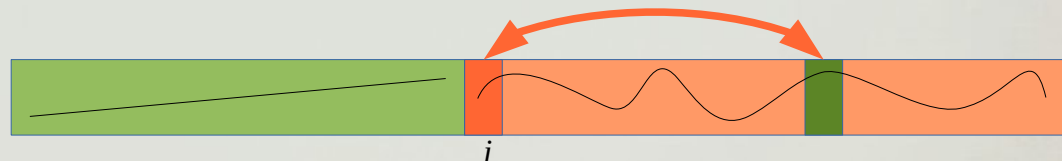
Navadno izbiranje

- Ideja algoritma
 - urejen seznam gradimo postopoma
 - na vsakem koraku poiščemo najmanjši element
 - in ga *dodamo* na konec urejenega dela
 - v tabeli: dodajanje → zamenjava
- Psevdokoda
- Sled



Navadno izbiranje

- Pravilnost algoritma
 - Zančna invarianta
 - Vsi elementi v urejenem delu so urejeni in manjši od elementov v neurejenem delu

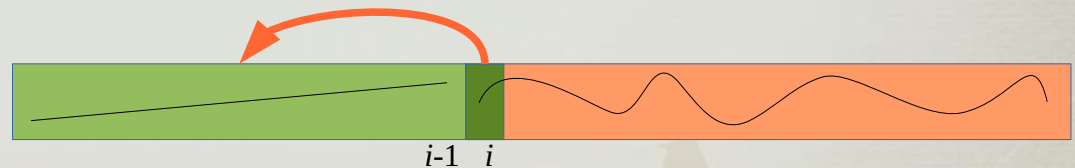


Navadno izbiranje

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

Navadno vstavljanje

- Ideja algoritma
 - urejen seznam gradimo postopoma
 - zaporedoma obdelujemo elemente in
 - jih *vstavljamo* na pravo mesto
- Psevdokoda
- 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

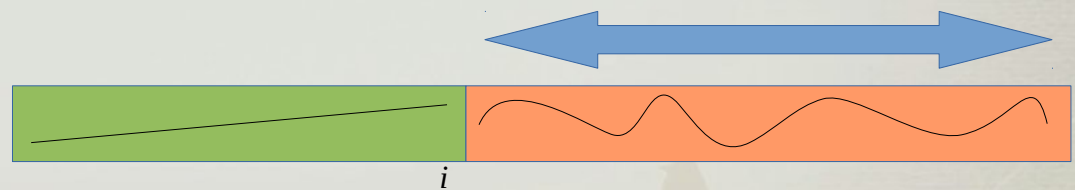
- Pravilnost algoritma
 - Zančna invarianta (zunanja zanka)
 - v i -ti iteraciji je tabela $a[0, 1, \dots, i-1]$ urejena
 - torej $a_0 < a_1 < \dots < a_{i-1}$
- Veljavnost
 - prej
 - med
 - potem

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
- Psevdokoda
- Sled

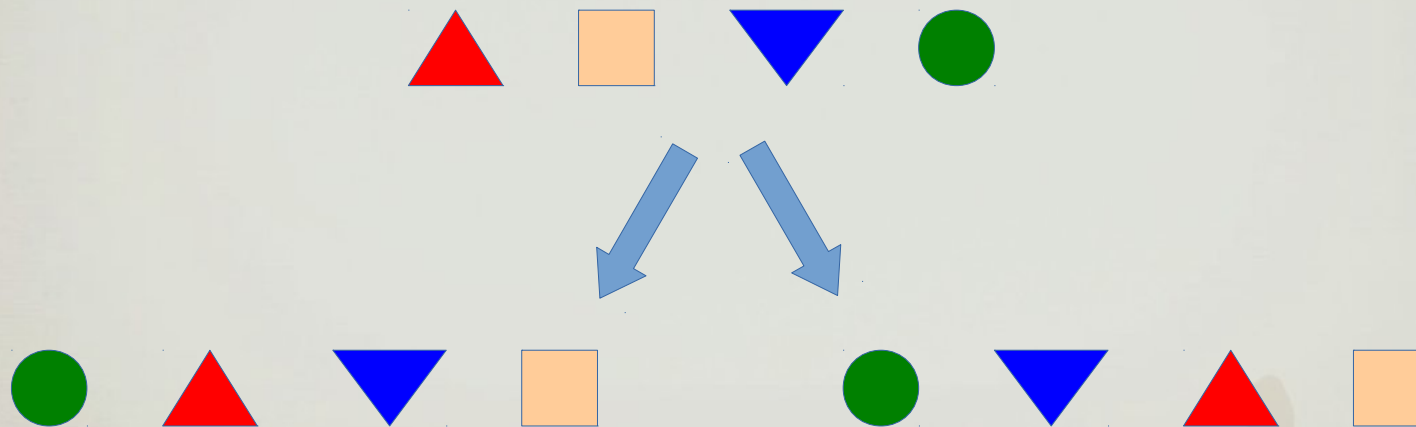


Navadne zamenjave

- 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

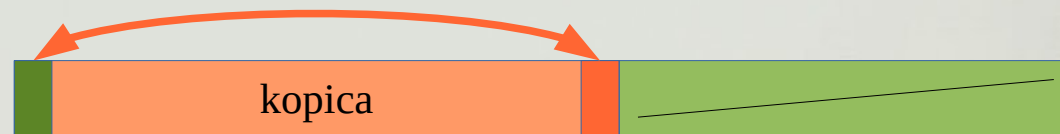
- Strukturirani podatki
 - urejanje po ključu
- Stabilnost
 - ohranja prvotni vrstni red pri enakih elementih



- Uporaba
 - urejanje po več ključih

Urejanje s kopico

- Ideja algoritma (*heapsort*)
 - uporabimo urejanje z izbiranjem
 - za iskanje največjega elementa uporabimo kopico
 - kopico zgradimo v prvem delu tabele
 - drugi del tabele je urejeni del
- Sled



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

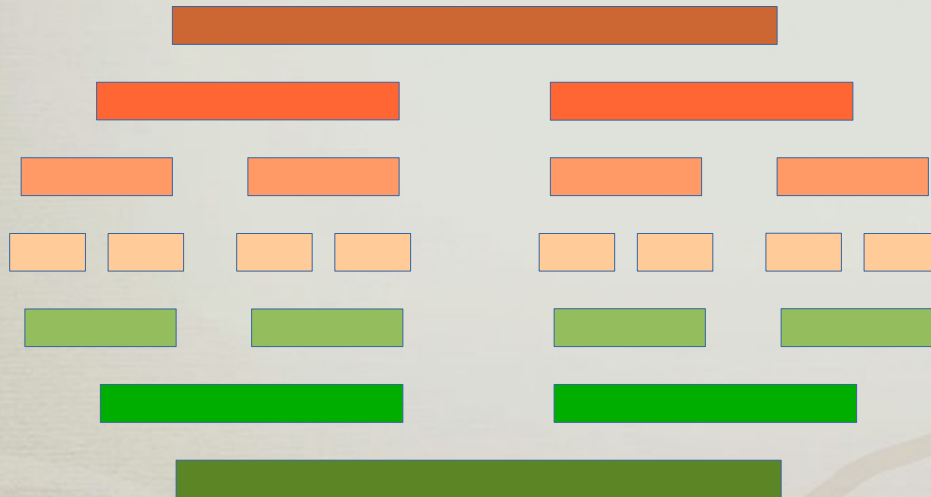


Urejanje z zlivanjem

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

delitev tabel

zlivanje

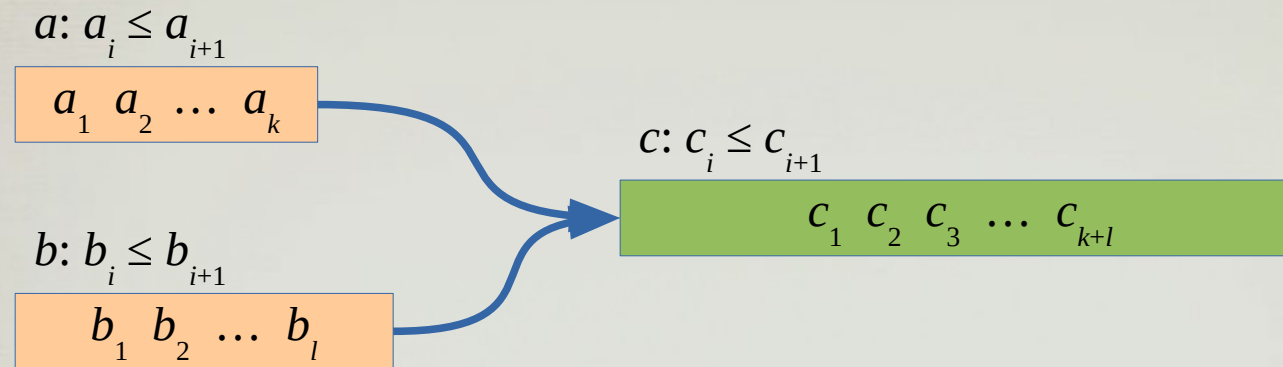


*Kdaj in kako
ustavimo
rekurzijo?*



Urejanje z zlivanjem

- Zlivanje urejenih podtabel



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

Urejanje z zlivanjem

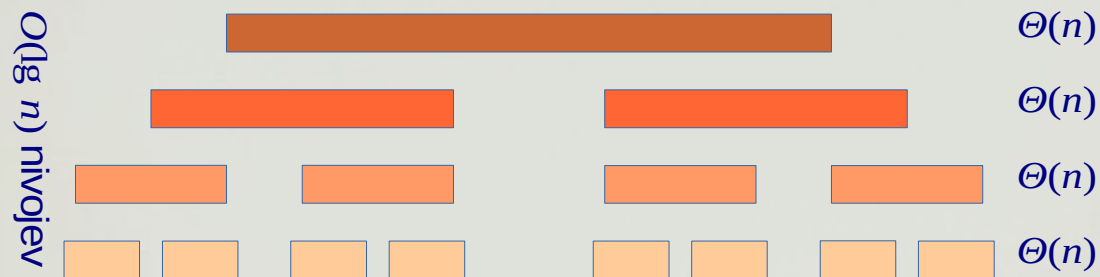
- Psevdokoda

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?



$O(n \lg n)$

Hitro urejanje

- Quicksort

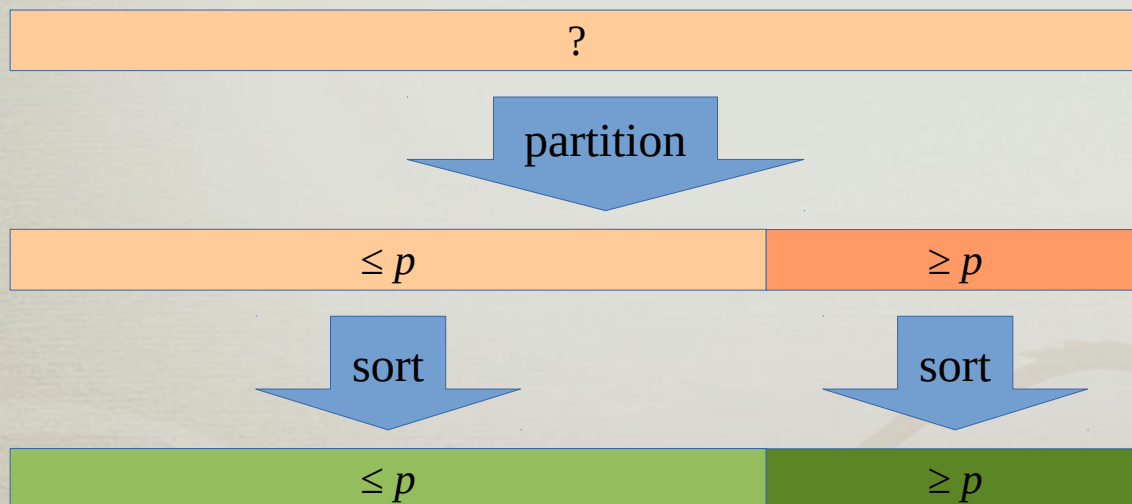
- verjetno najbolj pogosto uporabljen algoritem za urejanje
- splošen, uporablja primerjave
- algoritem deluje neposredno v tabeli
- potrebuje malo dodatnega prostora
- dobro deluje za različne vrste podatkov
- v povprečju zelo hiter
- previdno pri implementaciji



Quicksort, 1960

Hitro urejanje

- Ideja algoritma – deli & vladaj
 - Tabela 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



*Sestavljanje
rešitve je
trivialno.*

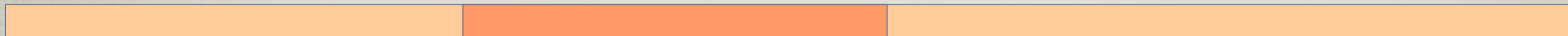


Hitro urejanje

- Porazdeljevanje
 - Ogromno načinov

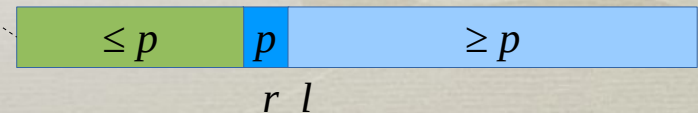
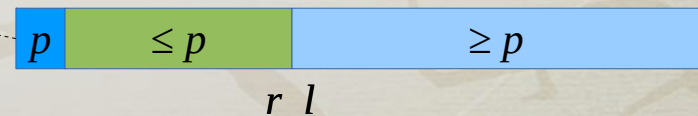
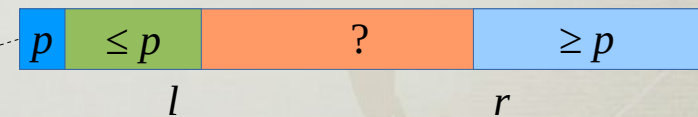
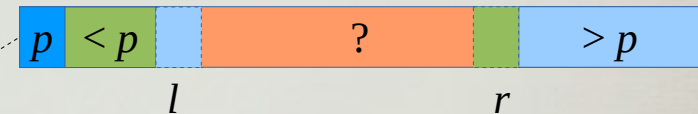
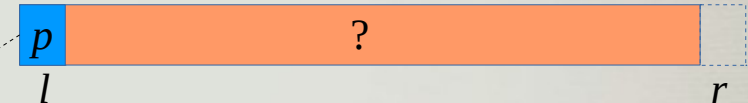
- izbira pivota
- inplace
- pogoji v zankah
- več pivotov

left right



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: $O(n \lg n)$



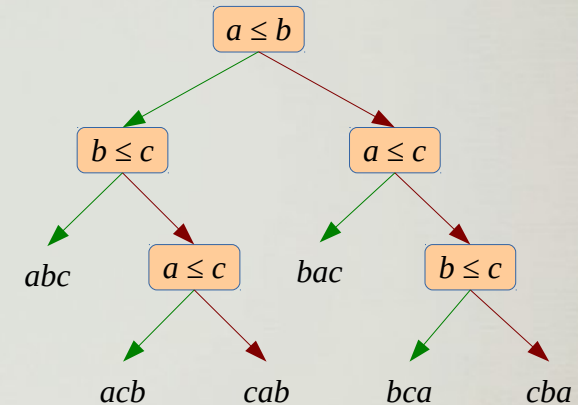
Inženiring algoritmov

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



Zahtevnost urejanja

- Spodnja meja zahtevnosti urejanja
 - vsaj koliko časa potrebuje katerikoli algoritem?
 - urejanje s primerjavami
 - ugotavljanje permutacije
 - permutiranje
- Odločitveno drevo
 - listi in notranja vozlišča
 - koliko je listov?
 - kakšna je višina drevesa?



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 (primerjave)	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 <i>in-place</i> , dodatni prostor
Hitro urejanje	$O(n^2)$, avg: $\Theta(n \log n)$	randomizacija, dodatni prostor