ALGORITMI I STRUKTURE PODATAKA – GRUPA 9
DODACI UZ ŠESTI TJEDAN NASTAVE
Robert Manger, 3. travnja 2007.

POSTUPCI SORTIRANJA

Problem (uzlaznog) sortiranja glasi ovako. Zadano je polje a[ ] duljine *n*. Vrijednosti elemenata mogu se uspoređivati dobrim uređajem <= . Treba permutirati vrijednosti elemenata tako da nakon permutiranja bude a[0] <= a[1] <= .... <= a[*n*-1] .

Sortiranje izborom najmanjeg elementa – selection sort

Opis. Prođe se poljem i pronađe se najmanji element. Zatim se najmanji element zamijeni s početnim. Dalje se promatra ostatak polja (bez početnog elementa) i ponavlja isti postupak.

Primjer.

Početno polje:	17	31	3	43	11	24	8
Nakon 1. zamjene:	3 ‖	31	17	43	11	24	8
Nakon 2. zamjene:	3	8 ‖	17	43	11	24	31
Nakon 3. zamjene:	3	8	11 ‖	43	17	24	31
Nakon 4. zamjene:	3	8	11	17 ‖	43	24	31
Nakon 5. zamjene:	3	8	11	17	24 ‖	43	31
Nakon 6. zamjene:	3	8	11	17	24	31 ‖	43

U svakom prolasku pronađeni najmanji element označen je sjenčanjem. Promatrani dio polja nalazi se s desne strane dvostruke okomite crte. Sedmi prolazak nije potreban jer je nakon šestog prolaska preostali dio polja duljine 1.

Analiza složenosti.
- U prvom prolasku imamo *n*-1 usporedbi, a u svakom idućem koraku broj usporedbi se smanji za 1. Dakle ukupni broj usporedbi iznosi:
  (*n*-1) + (*n*-2) + (*n*-3) + ... + 2 + 1 = *n*(*n*-1)/2.
- U svakom prolasku postoji još jedna zamjena, pa imamo još 3(n - 1) operacija pridruživanja.
- Dakle ukupni broj operacija je :
  $n(n-1)/2 + 3(n-1) = O(n^2)$.
- Algoritam obavlja isti posao bez obzira na početno stanje polja, čak i ako je polje već na početku bilo sortirano.

Sortiranje zamjenom susjednih elemenata – bubble sort

Opis. Prolazimo poljem od početka prema kraju i uspoređujemo susjedne elemente. Ako je neki element veći od sljedećeg elementa, zamijenimo im vrijednosti. Kad na taj način dođemo do kraja polja, najveća vrijednost doći će na posljednje mjesto. Nakon toga ponavljamo postupak na skraćenom polju (bez zadnjeg elementa). Algoritam se smije zaustaviti čim on u nekom prolazu ustanovi da nema parova elemenata koje bi trebalo zamijeniti.

Primjer.

Prvi prolaz:	17	31	3	43	11	24	8
	17	3	31	43	11	24	8
	17	3	31	11	43	24	8
	17	3	31	11	24	43	8
	17	3	31	11	24	8 ‖	43
Drugi prolaz:	17	3	31	11	24	8 ‖	43
	3	17	31	11	24	8 ‖	43
	3	17	11	31	24	8 ‖	43
	3	17	11	24	31	8 ‖	43
	3	17	11	24	8 ‖	31	43
Treći prolaz:	3	17	11	24	8 ‖	31	43
	3	11	17	24	8 ‖	31	43
	3	11	17	8 ‖	24	31	43
Četvrti prolaz:	3	11	17	8 ‖	24	31	43
	3	11	8 ‖	17	24	31	43
Peti prolaz:	3	11	8 ‖	17	24	31	43
	3	8 ‖	11	17	24	31	43
Šesti prolaz:	3	8 ‖	11	17	24	31	43

U svakom koraku dva susjedna elementa koji trebaju zamijeniti vrijednosti označeni su sjenčanjem. Promatrani dio polja nalazi se s lijeve strane dvostruke okomite crte. Šesti prolaz služi zato da se ustanovi da više nema parova susjednih elemenata koje bi trebalo zamijeniti.

Analiza složenosti.

- U prvom prolazu imamo u najgorem slučaju n-1 usporedbi i n-1 zamjena elemenata.
- U drugom prolazu imamo u najgorem slučaju n-2 usporedbi i n-2 zamjena elemenata.
- .....
- U (n-1)-vom prolazu imamo u najgorem slučaju 1 usporedbu i 1 zamjenu elemenata.
- Dakle ukupan broj operacija u najgorem slučaju iznosi:

$4(n\text{-}1) + 4(n\text{-}2) + ... + 4\times1 = 4n(n\text{-}1)/2 = 2n(n\text{-}1)$.

- Ocijenjeni broj operacija zaista se dobiva kad su elementi početnog polja sortirani silazno.
- S druge strane, ako je početno polje već sortirano uzlazno, obavit će se samo jedan prolaz s n-1 uspoređivanja i 0 zamjena.
- Za bilo koje drugo uređenje početnog polja broj operacija je između (n-1) i $2n(n$-1).
- U svakom slučaju, vrijeme izvođenja je O (n^2).

Sortiranje umetanjem – insertion sort

Opis. Za vrijeme rada algoritma, početni komad polja je već sortiran, a ostatak polja nije sortiran. U jednom prolasku algoritam uzima prvi element iz nesortiranog dijela. Te ga umetne na «pravo mjesto» (u smislu sortiranog redoslijeda) u sortirani dio, pri čemu dolazi do pomicanja nekih elemenata za jedno mjesto dalje. Dakle, jednim prolaskom duljina početnog sortiranog dijela se poveća za 1, a duljina nesortiranog dijela se smanji za 1.

Primjer.

Početno polje:	17	‖ 31	3	43	11	24	8
Nakon 1. prolaska:	17	31	‖ 3	43	11	24	8
Nakon 2. prolaska:	3	17	31	‖ 43	11	24	31
Nakon 3. prolaska:	3	17	31	43	‖ 11	24	8
Nakon 4. prolaska:	3	11	17	31	43	‖ 24	8
Nakon 5. prolaska:	3	11	17	24	31	43	‖ 8
Nakon 6. prolaska:	3	8	11	17	24	31	43 ‖

U svakom prolasku, element iz nesortiranog dijela polja koji se umeće u sortirani dio označen je sjenčanjem. Mjesto umetanja označeno je strelicom. Sortirani dio polja nalazi se s lijeve strane dvostruke okomite crte, a nesortirani s desne strane.

Analiza složenosti.

- U k-tom prolasku natraške prolazimo sortiranim dijelom polja duljine k. Zatečene elemente pomičemo za jedno mjesto dalje dok god su oni veći od elementa kojeg želimo umetnuti na «pravo mjesto». To u najgorem slučaju daje k usporedbi i otprilike isto toliko pridruživanja.
- Dakle ukupni broj operacija u najgorem slučaju je otprilike $2\times1 + 2\times2 + ... + 2(n\text{-}1) = n(n\text{-}1)$.
- Red veličine za vrijeme izvođenja je opet O (n^2).

Višestruko sortiranje umetanjem - Shell sort

Opis. Za zadani k promatra se k različitih pod-polja sastavljenih od elemenata originalnog polja međusobno udaljenih za točno k mjesta. Preciznije, za zadani k promatraju se pod-polja:

P_0 : a[0], a[k], a[2k], ...
P_1 : a[1], a[k+1], a[2k+1], ...
...
...
P_{k-1} : a[k-1], a[2k-1], a[3k-1], ...

Proces k-struke primjene insertion sort-a u svrhu zasebnog sortiranja svakog od ovih k pod-polja zove se k-subsort. Shell sort radi tako da za definirani padajući niz $k_1 > k_2 > \ldots > k_m=1$ obavi k_1 –subsort, zatim k_2–subsort, ... na kraju k_m–subsort. Promijetimo da zadnji k_m–subsort (ustvari 1-subsort) izvodi obični insertion sort na cijelom originalnom polju. Zato korektnost algoritma nije upitna. Prethodni k–subsortovi služe zato da (radeći nad kratkim pod-poljima) donekle urede cijelo polje tako da idući k–subsortovi imaju sve manje posla.

Primjer

Promatramo rad Shell sorta s padajućim nizom k-ova: 4 > 2 > 1.

Početno polje:	17	31	3	43	11	24	8
Ulaz za 4-subsort:	17	31	3	43	11	24	8
Izlaz iz 4-subsorta:	11	24	3	43	17	31	8
Ulaz za 2-subsort:	11	24	3	43	17	31	8
Izlaz iz 2-subsorta:	3	24	8	31	11	43	17
Ulaz za 1 subsort:	3	24	8	31	11	43	17
Izlaz iz 1-subsorta:	3	8	11	17	24	31	43

Unutar svakog retka ista boja označava elemente koji u okviru dotičnog k-subsorta čine jedno pod-polje.

Analiza složenosti.
- Prosječno vrijeme izvođenja je otvoreni problem.
- Ako je padajući niz $k_1 > k_2 > \ldots > k_m=1$ oblika $2^l-1 > 2^{l-1}-1 > \ldots > 7 > 3 > 1$, tada se može pokazati da je složenost u najgorem slučaju $O(n^{3/2})$.

Sažimanje sortiranih polja – merge

Sad rješavamo jedan pomoćni problem koji je usko vezan s problemom sortiranja. Zadana su dva uzlazno sortirana polja: a[ ] duljine *n* i b[ ] duljine *m*. Ta dva polja treba na što jednostavniji način prepisati u treće polje c[ ] duljine *n*+*m* koje je opet uzlazno sortirano.

Opis. Simultano čitamo a[ ] i b[ ] od početka do kraja, te istovremeno prepisujemo pročitane elemente u c[ ]. Znači, pamtimo kazaljku tekućeg elementa u a[ ] odnosno b[ ], te kazaljku na prvo slobodno mjesto u c[ ]. Uspoređujemo tekuće elemente u a[ ] i b[ ].

- Ako je jedan od tih tekućih elemenata manji, tada ga prepišemo u c[ ] te pomaknemo odgovarajuću kazaljku u a[ ] odnosno b[ ].
- Ako su tekući elementi jednaki, oba prepišemo u c[ ] te pomaknemo obje kazaljke u a[ ] i b[ ].
- Kod svakog prepisivanja pomičemo kazaljku u c[ ].

Kad jedno od polja a[ ] i b[ ] pročitamo do kraja, tada ostatak drugog polja izravno prepišemo u c[ ].

Primjer.

	Polje a[ ]	Polje b[ ]	Polje c[ ]
1. korak:	**3** 17 31 43	8 11 24	
2. korak:	3 17 31 43	**8** 11 24	3
3. korak:	3 17 31 43	8 **11** 24	3 8
4. korak:	3 **17** 31 43	8 11 24	3 8 11
5. korak:	3 17 31 43	8 11 **24**	3 8 11 17
6. korak:	3 17 **31 43**	8 11 24	3 8 11 17 24
Kraj:	3 17 31 43	8 11 24	3 8 11 17 24 31 43

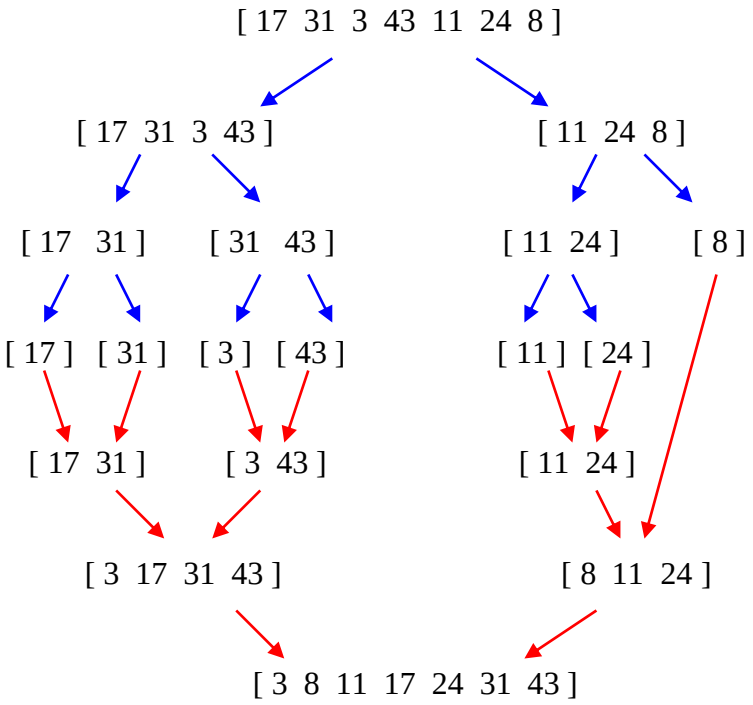
Strelice označavaju kazaljke tekućih elemenata u a[ ] i b[ ], odnosno kazaljku slobodnog mjesta u c[ ]. Sjenčanje označava elemente koji su u određenom koraku odabrani za prepisivanje.

Analiza složenosti. Očito je da je vrijeme izvršavanja algoritma sažimanja proporcionalno sa zbrojem duljina zadanih polja odnosno s duljinom rezultirajućeg polja, dakle O(*n*+*m*).

Sortiranje sažimanjem – merge sort

Opis. Riječ je o rekurzivnom algoritmu tipa «podijeli pa vladaj». Zadano polje podijeli se na dva manja polja podjednakih duljina. Ta dva manja polja zasebno se sortiraju rekurzivnim pozivima istog algoritma. Zatim se mala sortirana polja sažimlju u jedno (sortirano) uz pomoć prethodno opisanog postupka sažimanja – merge.

Primjer.



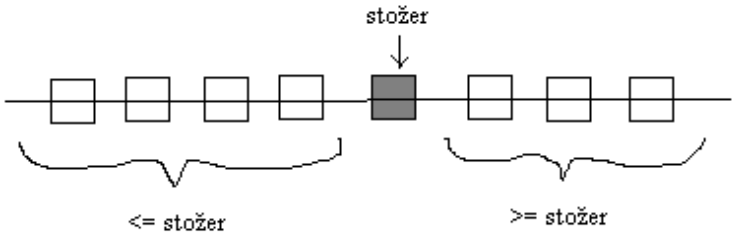
Uglate zagrade ovdje označavaju pojedina polja koja nastaju u postupku sortiranja. Plave strelice označavaju podjele većih polja na manja. Crvene strelice označavaju sažimanje manjih sortiranih polja u veća.

Analiza složenosti.

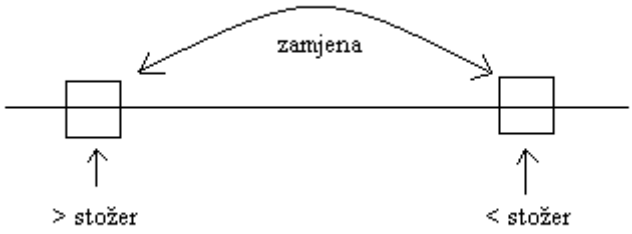
- Svaki rekurzivni poziv algoritma ima vrijeme računanja proporcionalno duljini rezultirajućeg (sažetog) polja kojeg će on proizvesti.
- Skup svih rekurzivnih poziva koji nastaju pri rješavanju polaznog problema može se prikazati binarnim stablom.
- Kad gledamo sve rekurzivne pozive na istom nivou stabla, primjećujemo da oni rade s poljima čiji zbroj duljina je jednak duljini polaznog polja n. Dakle zbrojeno vrijeme računanja za sve rekurzivne pozive na istom nivou je $O(n)$.
- Budući da nivoa ima $\log_2 n+1$, ukupna složenost algoritma je $(\log_2 n+1)\times O(n) = O(n \log n)$.
- Mana algoritma je korištenje dodatne memorije zbog prepisivanja polja.
- Dobra osobina algoritma je mogućnost primjene u situaciji kad podaci ne stanu u glavnu memoriju već su pohranjeni u (sekvencijalnoj) datoteci.

Brzo sortiranje – quick sort

Opis. Opet je riječ o rekurzivnom algoritmu tipa «podijeli pa vladaj». Odabere se jedan element u polju, takozvani stožer. Svi ostali elementi razvrstaju se ispred (lijevo od) odnosno iza (desno od) stožera ovisno o tome da li su <= ili >= od stožera.



Očigledno, nakon ovakvog razvrstavanja stožer se nalazi na svom «konačnom mjestu» u smislu traženog sortiranog poretka. Da bi se polje sortiralo do kraja, dovoljno je zasebno (neovisno) sortirati pod-polje lijevo odnosno desno od stožera. Sortiranje pod-polja postiže se rekurzivnim pozivima istog algoritma, ili na trivijalni način za pod-polje duljine 0 ili 1.



Samo razvrstavanje elemenata lijevo odnosno desno od stožera efikasno se obavlja tako da pomičemo kazaljku s lijevog odnosno desnog kraja polja, sve dok su tekući elementi <= odnosno >= od stožera. Kad se obje kazaljke zaustave, našli smo element slijeva koji je > stožera i element s desna koji je < stožera. Ta dva elementa na «krivim stranama» međusobno zamijenimo i zatim nastavimo s pomicanjem kazaljki. Razvrstavanje je završeno kad se kazaljke prekriže.

Primjer.

Početno polje; stožer je početni element:	17	31	3	43	11	24	8
Pomicanje i zaustavljanje kazaljki:	17	31	3	43	11	24	8
		$\uparrow_L$					$\uparrow_D$
Zamjena elemenata na koje pokazuju kazaljke:	17	8	3	43	11	24	31
		$\uparrow_L$					$\uparrow_D$
Pomicanje i zaustavljanje kazaljki:	17	8	3	43	11	24	31
				$\uparrow_L$	$\uparrow_D$		
Zamjena elemenata na koje pokazuju kazaljke:	17	8	3	11	43	24	31
				$\uparrow_L$	$\uparrow_D$		
Pomicanje i zaustavljanje kazaljki:	17	8	3	11	43	24	31
				$\uparrow_D$	$\uparrow_L$		
Kazaljke su se prekrižile. Ubacujemo stožer:	[8	3	11]	17	[43	24	31]
Dva pod-polja sortiramo rekurzivnim pozivima:	[3	8	11]	17	[24	31	43]

Simboli $\uparrow_L$ odnosno $\uparrow_D$ označavaju položaj kazaljke s lijevog odnosno desnog kraja polja u raznim fazama razvrstavanja elemenata. Dvostruka okomita crta pokazuje da je stožer je izdvojen tijekom postupka rasvrstavanja. Uglate zagrade zatvaraju pod-polja koja se sortiraju rekurzivnim pozivima istog algoritma. Nisu prikazani detalji unutar tih rekurzivnih poziva.

Analiza složenosti.

- Može se lako provjeriti da algoritam u najgorem slučaju ima složenost $\Theta(n^2)$. Najgori slučaj nastupa kad je stožer početni element i kad je polje već sortirano.
- Postoji matematički dokaz da je prosječno vrijeme izvršavanja $\Theta(n \log n)$.
- Praktična iskustva (a posteriori analiza) pokazuju da je algoritam u stvarnosti izuzetno brz, dakle brži nego što se to može prepoznati iz teorijskih ocjena.
- Ponašanje algoritma u priličnoj mjeri ovisi o načinu biranja stožera. Često korištene mogućnosti za izbor stožera su: početni element (kao u našem primjeru); medijan izabran između tri elementa koji se nalaze na početku, kraju odnosno sredini polja.