1 Asimptoticna Notacija

Naj bo dana funkcija $g : N \to N$, potem funkcijo $f : N \to N$ pisemo:

$f(n) = \mathcal{O}(g(n))$, ce $\exists c > 0$, da je $\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq c$.
oz. $f(n) = \mathcal{O}(g(n))$, ce $\exists c > 0, n_0 > 0 \forall n \geq n_0 : f(n) \leq cg(n)$.
$\Rightarrow$ sklepamo, da f narasca **kvecjemu tako hitro** kot g.

$f(n) = \Omega(g(n))$, ce $\exists c > 0$, da je $c \leq \lim_{n\to\infty} \frac{f(n)}{g(n)}$.
oz. $f(n) = \Omega(g(n))$, ce $\exists c > 0, n_0 > 0 \forall n \geq n_0 : cg(n) \leq f(n)$.
$\Rightarrow$ sklepamo, da f narasca **vsaj tako hitro** kot g.

$f(n) = \Theta(g(n))$, ce $\exists c_1, c_2.c_2 > 0$, da je $c_1 \leq \lim_{n\to\infty} \frac{f(n)}{g(n)} \leq c_2$.
oz. $f(n) = \Theta(g(n))$, ce $\exists c_1, c_2 > 0, n_0 > 0 \forall n \geq n_0 : c_1 g(n) \leq f(n) \leq c_2 g(n)$. $\Rightarrow$ sklepamo, da f narasca **podobno hitro** kot g.

$f(n) = o(g(n))$, ce je $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.
oz. $f(n) = o(g(n))$, ce $\forall c > 0, \exists n_0 > 0 \forall n \geq n_0 : f(n) < cg(n)$.
$\Rightarrow$ sklepamo, da f narasca **pocasneje** kot g.

$f(n) = \omega(g(n))$, ce je $\lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty$.
oz. $f(n) = \omega(g(n))$, ce $\forall c > 0, \exists n_0 > 0 \forall n \geq n_0 : cg(n) < f(n)$.
$\Rightarrow$ sklepamo, da f narasca **hitreje** kot g.

2 Urejanje

2.1 Urejanje s kopico

Gre za nestabilen sortirni algoritem. Operacije:

Vstavljanje: Visina drevesa je h. Element vstavimo na zadnji nivo, k prvemu prostemu listu. V najslabsem primeru moramo popravljati navzgor do korena. $\mathcal{O}(log_2 n)$.

Odstranjevanje: Odstranimo korenski element in ga zamenjamo s skrajno desnim otrokom, na zadnjem nivoju. Element, katerega smo ustavili v koren popravljamo v najslabsem primeru spet do najnizjega nivoja. $\mathcal{O}(log_2 n)$.

Ustvarjanje kopice iz podane tabele: Sestavis kopico iz podane tabele. Najnizjega nivoja ne tikas. Nato se sprehodis po vseh nivojih navzgor iz desne proti levi, pa popravljas kopico navzdol. $\mathcal{O}(n)$.

2.2 Hitro urejanje

Gre za nestabilen sortirni algoritem. Za pivotni element obicajno izberemo najbol levi element v tabeli.

Psevdokoda: todo ko izvemo kateri algo je pravilen

2.2.1 Casovna zahtevnost

V najslabsem primeru izberemo prvi element za pivotni in dobimo ze urejeno tabelo. Tako se z indeksom i na vsakem nivoju sprehodimo do konca tabele (V rekurziji dobimo izrojeno drevo visine n). Iz tega sledi $\mathcal{O}(n^2)$. V splosnem pa za quicksort velja casovna zahtevnost $\Theta(nlog_2 n)$.

2.3 Urejanje z zlivanjem

Gre za stabilni sortirni algoritem. Tabelo najprej razdeljujemo na $\lceil$polovico $\rceil$ dolzine tabele. Ko pridemo do konca se ustavimo in zacnemo urejati navzgor po drevesu.

2.3.1 Casovna zahtevnost

Vedno $\mathcal{O}(nlog_2 n)$, saj tabelo razdeljujemo na polovico, tako da se rekurzivno drevo ne mora izroditi.

2.4 Urejanje s stetjem

Gre za stabilni sortirni algoritem $\mathcal{O}(n)$. Imejmo tabelo $[2, 1, 0, 0, 1, 2, 1]$, prestejemo pojavitve stevil $0, 1$, in 2. Ter jih zapisemo v dodatno tabelo.

$[2, 3, 2] \to$ cumsum $\to [2, 5, 7] \to$ popravek indeksov $\to [1, 4, 6]$. Potem ustvarimo novo tabelo velikosti originalne tabele. Urejanje zacnemo tako, da se sprehodimo po originalni tabeli (od zadaj proti zacetku)

In pogledamo v tabelo s kumulativno vsoto, kam se slika trenutni element, ter stevilo v kumilativni vsoti zmanjsamo za 1. Postopek ponavljamo.

2.5 Korensko urejanje

Gre za stabilni sortirni algoritem $\mathcal{O}(n)$. Urejas samo po stevkah. Primer sledi izvajanja:

$a = [36, 12, 27, 17]$

$a_e = [12, 36, 27, 17]$ (sortiramo po enicah)

$a_d = [12, 17, 27, 36]$ (sortiramo po deseticah)

3 Deli in vladaj

Problem razdelimo na vec **enakih** podproblemov.

3.1 Masters Theorem

$$T(n) = aT(\tfrac{n}{b}) + \mathcal{O}(n^d)$$

a - stevilo delitev problema

b - faktor deljenja problema

d - Zahtevnost zdruzevanja problemov

3.1.1 Ocena casovne zahtevnosti algoritma:

1. $a < b^d \Rightarrow T(n) = \mathcal{O}(n^d)$
2. $a = b^d \Rightarrow T(n) = \mathcal{O}(n^d \log_2 n)$
3. $a > b^d \Rightarrow T(n) = \mathcal{O}(n^{log_b a})$

3.2 Naivni algoritem za mnozenje stevil

Stevili a in b delimo na polovico, dokler ne pridemo do same stevke. $a = [a_1, a_0]$, $b = [b_1, b_0]$. n je stevilo stevk v posamezni iteraciji.

$$ab = 10^n a_1 b_1 + 10^{\frac{n}{2}}(a_0 b_1 + a_1 b_0) + a_0 b_0$$

3.3 Karacubov algoritem

Gre za izboljsavo naivnega mnozenja, saj potrebujemo mnoziti samo 3x.

$c_0 = a_0 b_0$

$c_1 = (a_1 + a_0)(b_1 + b_0) - c_0 - c_2$

$c_2 = a_1 b_1$

$$ab = 10^n c_0 + 10^{\frac{n}{2}} c_1 + a_1 b_1$$