Algoritmi in podatkovne strukture 2

Uvod v predmet

Poglavje 1: Računska zahtevnost

Luka Fürst

Predavanja

- Luka Fürst
- `luka.fuerst@fri.uni-lj.si`
- Govorilne ure
  - po dogovoru

Vaje

- Jure Savnik
- Rok Gomišček
- Luka Fürst

Režim

- Sprotno delo
  - kvizi in domače naloge
  - možnih 20 točk
- Pisni izpit
  - možnih 80 točk
- Ustni izpit
- Pogoji
  - vsaj 40 točk na pisnem izpitu
  - vsaj 50 točk s sprotnega dela in pisnega izpita skupaj
  - opravljen ustni izpit
- Ocena
  - temelji na seštevku točk sprotnega dela in pisnega izpita
  - po standardni lestvici

Sprotno delo

- Vsak teden prejmete kviz in domačo nalogo
- Kviz
  - kratke, enostavne naloge za preverjanje razumevanja osnov
- Domača naloga
  - programerska naloga
  - preverjanje s testnimi primeri, a ne vedno!

Okviren program

1. Računska zahtevnost
2. Metode načrtovanja algoritmov
3. Algoritmi na grafih
4. Algoritmi na nizih
5. Računska geometrija
6. »Razno«

- Pridržujem si pravico do sprememb!

Literatura (osnovna)

- Te prosojnice niso literatura!
- Thomas H. Cormen, Charles E. Leiserson, Ronald R. Rivest, Clifford Stein. Introduction to Algorithms, 4th Edition. MIT Press, 2022.
- Jon Kleinberg, Éva Tardos. Algorithm Design. Pearson, 2005.
- Dan Gusfield. Algorithms on Strings, Trees, and Sequences. Cambridge University Press, 1997.
- Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars. Computational Geometry, 3rd Edition. Springer, 2008.
- Steven Halim, Felix Halim, Suhendry Effendy. Competitive Programming 4. 2018.

Literatura (dodatna)

- Steven Skiena. The Algorithm Design Manual, 3rd Edition. Springer, 2020.
- Alfred Aho, John E. Hopcroft, Jeffrey D. Ullman. Data Structures and Algorithms. Pearson, 1983.
- Igor Kononenko. Načrtovanje podatkovnih struktur in algoritmov. Založba FRI, 1996.
- Boštjan Vilfan. Osnovni algoritmi. Založba FRI, 2002.
- Robert Sedgewick, Kevin Wayne. Algorithms, 4th Edition. Addison-Wesley, 2011.
- Aditya Y. Bhargava, Grokking Algorithms, 2nd Edition. Manning, 2024.

Cilj (gotovo ne letos)

- Nekoč, morda, nekje, nekdo . . .[1]
- Lastna knjiga!

[1] Tabu: Nekoč nekje

Problem in algoritem

- Problem
  - naloga, ki jo rešujemo s pomočjo računalnika
  - opis veljavnih vhodov
  - opis pričakovanih izhodov
- Algoritem
  - računski postopek, ki za vsak veljaven vhod v končnem času proizvede pričakovani izhod

Primer

- Problem
  - vhod: tabela A z n celimi števili
  - izhod: ista tabela A, a naraščajoče urejena
- Algoritmi
  - navadno vstavljanje
    - v i-ti iteraciji (za $i = 1, \ldots, n-1$) vstavi element $A[i]$ v že urejeno podtabelo $A[0 : i-1]$
  - urejanje z zlivanjem
    - razdeli tabelo na dva približno enaka dela
    - rekurzivno uredi vsakega posebej in urejena dela zlij
  - bogosort
    - če je tabela urejena, končaj
    - sicer naključno premešaj elemente in ponovi postopek

Računska zahtevnost

- Algoritme ponavadi vrednotimo po računski zahtevnosti
  - Koliko časa porabijo?
  - Koliko pomnilniškega prostora porabijo?
- Nejasnost št. 1
  - v najboljšem primeru?
  - v najslabšem primeru?
  - v »povprečnem« primeru?
- Nejasnost št. 2
  - na katerem računalniku?
  - v katerem programskem jeziku?
  - s kakšno implementacijo?

Nejasnost št. 1

- Ponavadi nas najbolj zanima najslabši primer
- Kljub temu se pogosto ukvarjamo tudi s povprečnim in najboljšim primerom
  - hitro urejanje (quicksort) v najslabšem primeru teče v času $\Theta(n^2)$, v povprečju (in pravzaprav v ogromni večini primerov) pa v času $\Theta(n \log n)$

Nejasnost št. 2

- Vpliva vse troje (računalnik, programski jezik, implementacija), a pri večini algoritmov se najbolj pozna stopnja naraščanja pri naraščanju velikosti vhoda
- Kako poraba časa oz. prostora narašča pri povečevanju velikosti vhoda?
- Nejasnost št. 3
  - kaj je velikost vhoda?

Velikost vhoda

- Podatek, od katerega je časovna zahtevnost ponavadi najbolj odvisna
- Urejanje
  - n (dolžina tabele)
- Algoritmi na grafih
  - n (število vozlišč)
  - m (število povezav)
- Včasih upoštevamo tudi nekatere druge mere zahtevnosti vhoda
  - npr. število čet (že urejenih podzaporedij) pri nekaterih algoritmih urejanja

Asimptotična računska zahtevnost

- n: velikost vhoda
- $f : \mathbb{N} \to \mathbb{R}^+$: poraba časa/prostora v odvisnosti od vhoda
- $O(f(n))$: množica vseh funkcij $\mathbb{N} \to \mathbb{R}^+$, ki od neke točke naprej naraščajo kvečjemu tako hitro kot funkcija $f(n)$

$$f(n) \in O(g(n)) \iff \exists c, n_0 : \forall n \geq n_0 : f(n) \leq cg(n)$$

- $O(n^2) = \{n^2,\ 6n^2 + 7n + 10,\ 1000n^2,\ 10n \lg n,\ n,\ \ldots\}$
- Zanemarimo koeficient pri vodilnem členu in vse počasneje naraščajoče člene

$O(.)$, $\Theta(.)$ in $\Omega(.)$

- $O(.)$: asimptotična zgornja meja

$$f(n) \in O(g(n)) \iff \exists c, n_0 : \forall n \geq n_0 : f(n) \leq cg(n)$$

- $\Omega(.)$: asimptotična spodnja meja

$$f(n) \in \Omega(g(n)) \iff \exists c, n_0 : \forall n \geq n_0 : f(n) \geq cg(n)$$

- $\Theta(.)$: asimptotično tesna meja

$$f(n) \in \Theta(g(n)) \iff \exists c_1, c_2, n_0 : \forall n \geq n_0 : \\ c_1 g(n) \leq f(n) \leq c_2 g(n)$$

Alternativne definicije

$$f(n) \in O(g(n)) \iff 0 \leq \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

$$f(n) \in \Omega(g(n)) \iff 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} \leq \infty$$

$$f(n) \in \Theta(g(n)) \iff 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

$o(.)$ in $\omega(.)$

- $o(f(n)) = O(f(n)) \setminus \Theta(f(n))$
- $\omega(f(n)) = \Omega(f(n)) \setminus \Theta(f(n))$

- Alternativni definiciji

$$f(n) \in o(g(n)) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

$$f(n) \in \omega(g(n)) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

Primerjava funkcij in števil

V svetu funkcij	V svetu števil
$f(n) \in O(g(n))$	$a \leq b$
$f(n) \in \Omega(g(n))$	$a \geq b$
$f(n) \in \Theta(g(n))$	$a = b$
$f(n) \in o(g(n))$	$a < b$
$f(n) \in \omega(g(n))$	$a > b$

Vsi sledeči stavki so pravilni

- Navadno vstavljanje v najslabšem primeru teče v času
  - $\Theta(n^2)$ (najbolj natančno)
  - $O(n^2)$
  - $\Omega(n^2)$
  - $O(n^3)$
  - $\Omega(n)$
  - $o(n^3)$
  - $\omega(n)$

Vsi sledeči stavki so pravilni

- Navadno vstavljanje
  - v vseh primerih teče v času $O(n^2)$
  - v vseh primerih teče v času $\Omega(n)$
  - v najslabšem primeru teče v času $\Theta(n^2)$
  - v najboljšem primeru teče v času $\Theta(n)$
  - v povprečnem primeru teče v času $\Theta(n^2)$
  - teče v času $O(n^2)$

Zloraba notacije

- Namesto $f(n) \in O(g(n))$ običajno pišemo $f(n) = O(g(n))$
- Precej huda zloraba, ki pa ima svoj smisel
- »Algoritem teče v času $T(n) = \Theta(n^2)$«
  - $T(n)$: neka kvadratna funkcija
- »Algoritem teče v času $T(n) = 5n^2 + \Theta(n)$«
  - $T(n) = 5n^2 + f(n)$
  - $f(n)$: neka linearna funkcija
- »Algoritem teče v času $T(n) = 5n^2 + O(n)$«
  - $T(n) = 5n^2 + f(n)$
  - $f(n)$: neka funkcija, ki narašča kvečjemu linearno

Nekaj lastnosti

- $\Theta(\log_a n) = \Theta(\log_b n) = \Theta(\log n)$
  - velja tudi za $O(.)$, $\Omega(.)$, $o(.)$ in $\omega(.)$
- $a^n = o(b^n)$, če $a < b$
- $n^a = o(b^n)$ za vsak $a \geq 0$ in $b > 1$
- $\log^k n = o(n)$ za vsak $k \geq 0$
- $\log n! = \Theta(n \log n)$
  - Stirlingova aproksimacija: $n! \approx (\frac{n}{e})^n \sqrt{2\pi n}$

Analiza navadnega vstavljanja

- Analizirajmo najslabši primer (padajoče urejena tabela)
- Kolikokrat se izvedejo elementarne operacije?
  - prirejanje celoštevilski spremenljivki
  - aritmetične in logične operacije
  - primerjava dveh celih števil
  - »režijski stroški« zank in pogojnih stavkov
  - ...
- Za začetek bodimo natančni ...

Analiza navadnega vstavljanja

function NAVADNO-VSTAVLJANJE(A)

$n \leftarrow |A|$ ▷ c_0

for $i \leftarrow 1 : n-1$ **do** ▷ $c_1 n$

$ključ \leftarrow A[i]$ ▷ $c_2(n-1)$

$j \leftarrow i-1$ ▷ $c_3(n-1)$

while $j \geq 0 \wedge A[j] > ključ$ **do** ▷ $c_4(i+1)$ $za\ i = 1 : n-1$

$A[j+1] \leftarrow A[j]$ ▷ $c_5 i$ $za\ i = 1 : n-1$

$j \leftarrow j-1$ ▷ $c_6 i$ $za\ i = 1 : n-1$

$A[j+1] \leftarrow ključ$ ▷ $c_7(n-1)$

$$T(n) = c_0 + c_1 n + (c_2 + c_3 + c_7)(n-1) + \sum_{i=1}^{n-1}(c_4(i+1) + c_5 i + c_6 i)$$

$$= (\frac{c_4}{2} + \frac{c_5}{2} + \frac{c_6}{2})n^2 + (c_1 + c_2 + c_3 + \frac{c_4}{2} - \frac{c_5}{2} - \frac{c_6}{2} + c_7)n$$

$$+ (c_0 - c_2 - c_3 - c_4 - c_7)$$

Analiza navadnega vstavljanja

- Konstante lahko združimo

$$T(n) = C_1 n^2 + C_2 n - C_3$$

- Vse razen vodilnega člena lahko zanemarimo

$$T(n) = C_1 n^2 + O(n)$$

- Koeficient vodilnega člena lahko zanemarimo

$$T(n) = \Theta(n^2)$$

Analiza navadnega vstavljanja

- V najslabšem primeru
  - tabela je padajoče urejena
  - $\Theta(n^2)$
- V najboljšem primeru
  - tabela je naraščajoče urejena
  - $\Theta(n)$
- V povprečnem primeru
  - notranja zanka se izvede $(i/2)$-krat
  - $\Theta(n^2)$

Analiza urejanja z zlivanjem

function UREDI-Z-ZLIVANJEM(A)
    UREDI-Z-ZLIVANJEM(A, 0, $|A| - 1$)

function UREDI-Z-ZLIVANJEM(A, p, r)
    if $r - p \geq 1$ **then**
        $q \leftarrow \lfloor (p + r) \,/\, 2 \rfloor$
        UREDI-Z-ZLIVANJEM(A, p, q)
        UREDI-Z-ZLIVANJEM($A, q + 1, r$)
        ZLIJ(A, p, q, r)

Analiza urejanja z zlivanjem

- $T(n)$: poraba časa za tabelo dolžine n
  - neobvezna poenostavitev: $n = 2^k$
- Tabelo razbijemo na dva enaka dela: $\Theta(1)$
- Vsak del uredimo z istim postopkom: $2T(n/2)$
- Urejena dela zlijemo: $\Theta(n)$

$$T(n) = \begin{cases} d, & \text{če } n = 1 \\ 2T(n/2) + cn, & \text{če } n \geq 2 \end{cases}$$

Analiza urejanja z zlivanjem

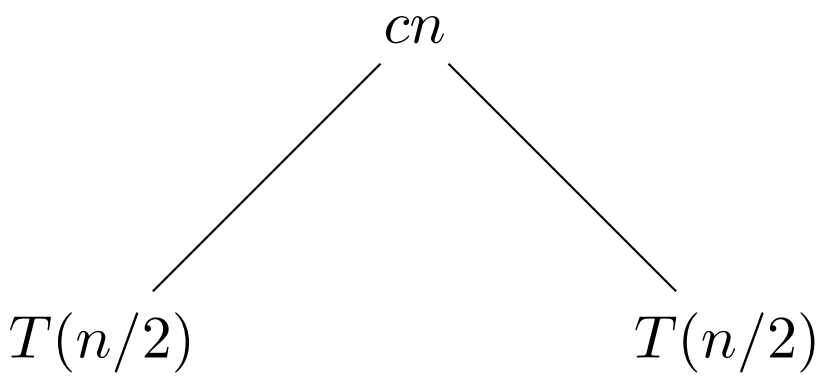
- Obstaja več načinov za reševanje tovrstnih rekurenčnih enačb
- Ena možnost: vstavljaj izraz $T(n)$ samega vase

$$
\begin{aligned}
T(n) &= cn + 2T(n/2) \\
T(n/2) &= cn/2 + 2T(n/4) \\
T(n/4) &= cn/4 + 2T(n/8) \\
&\dots
\end{aligned}
$$

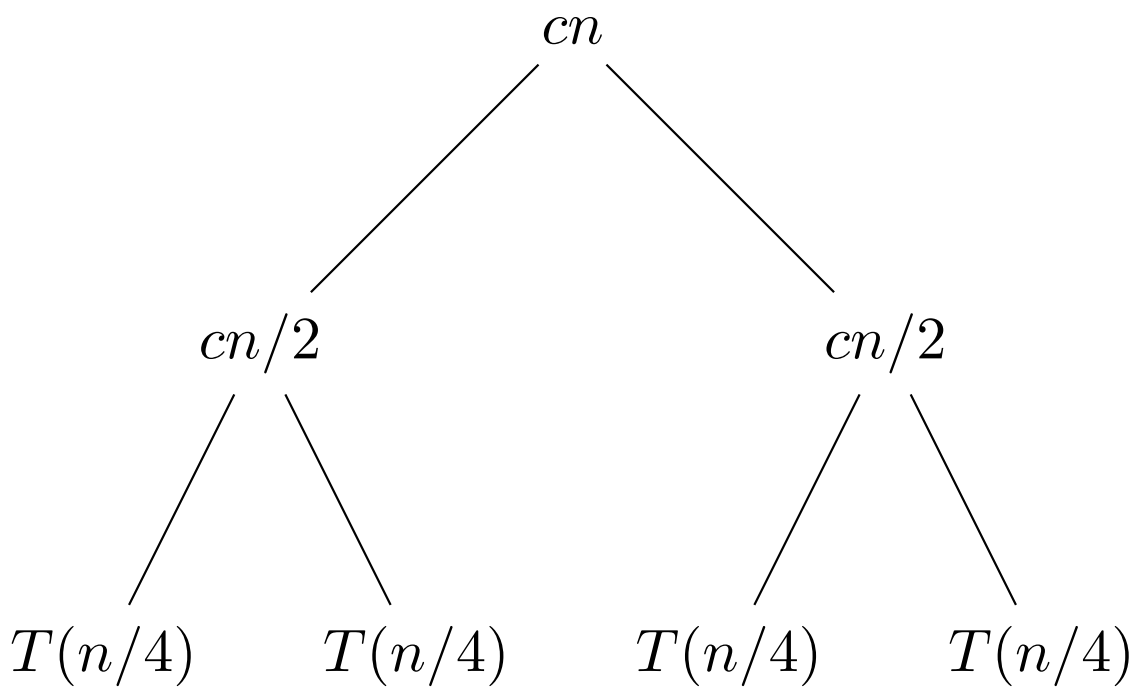
Analiza urejanja z zlivanjem

$$T(n)$$

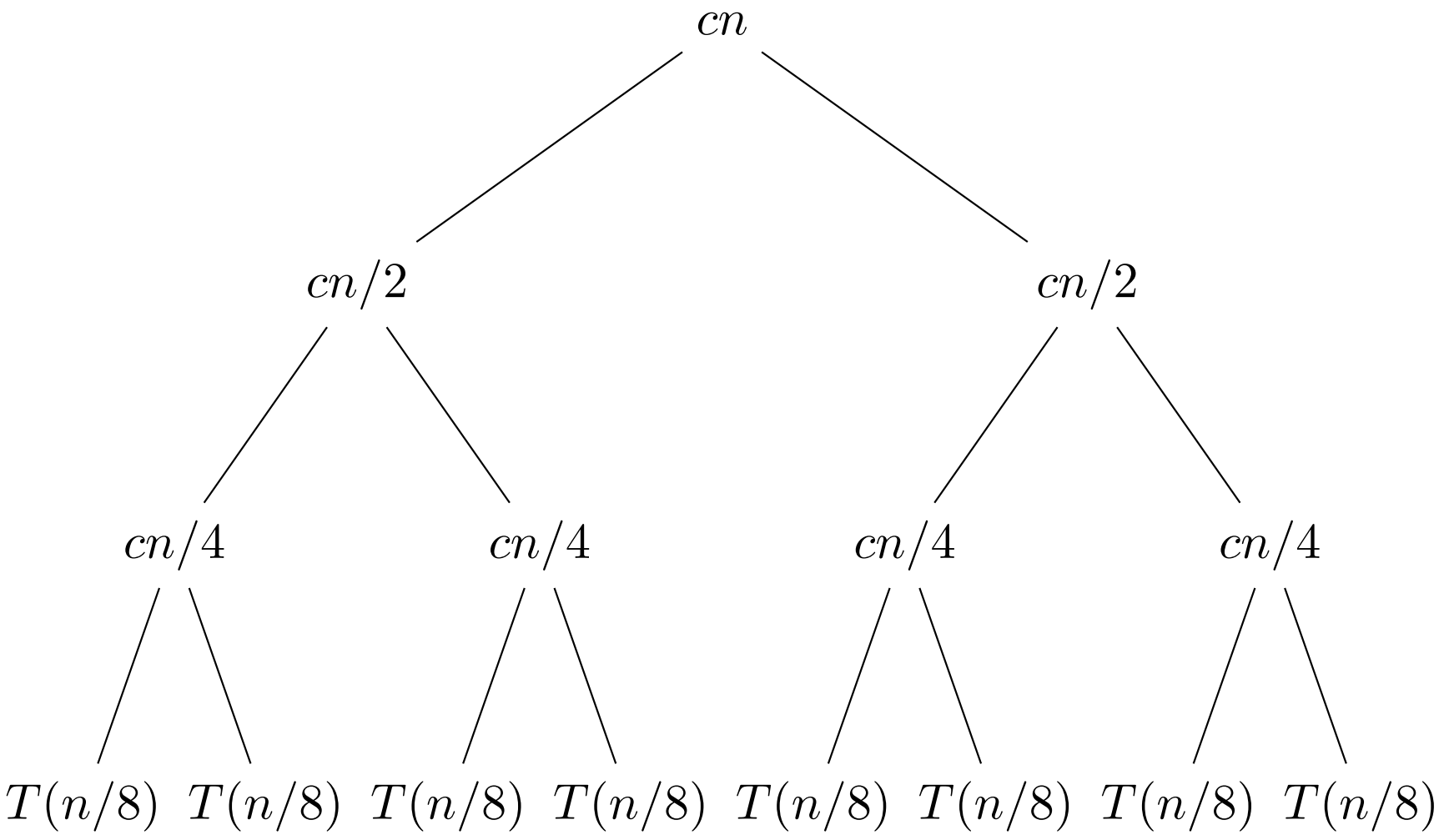
Analiza urejanja z zlivanjem



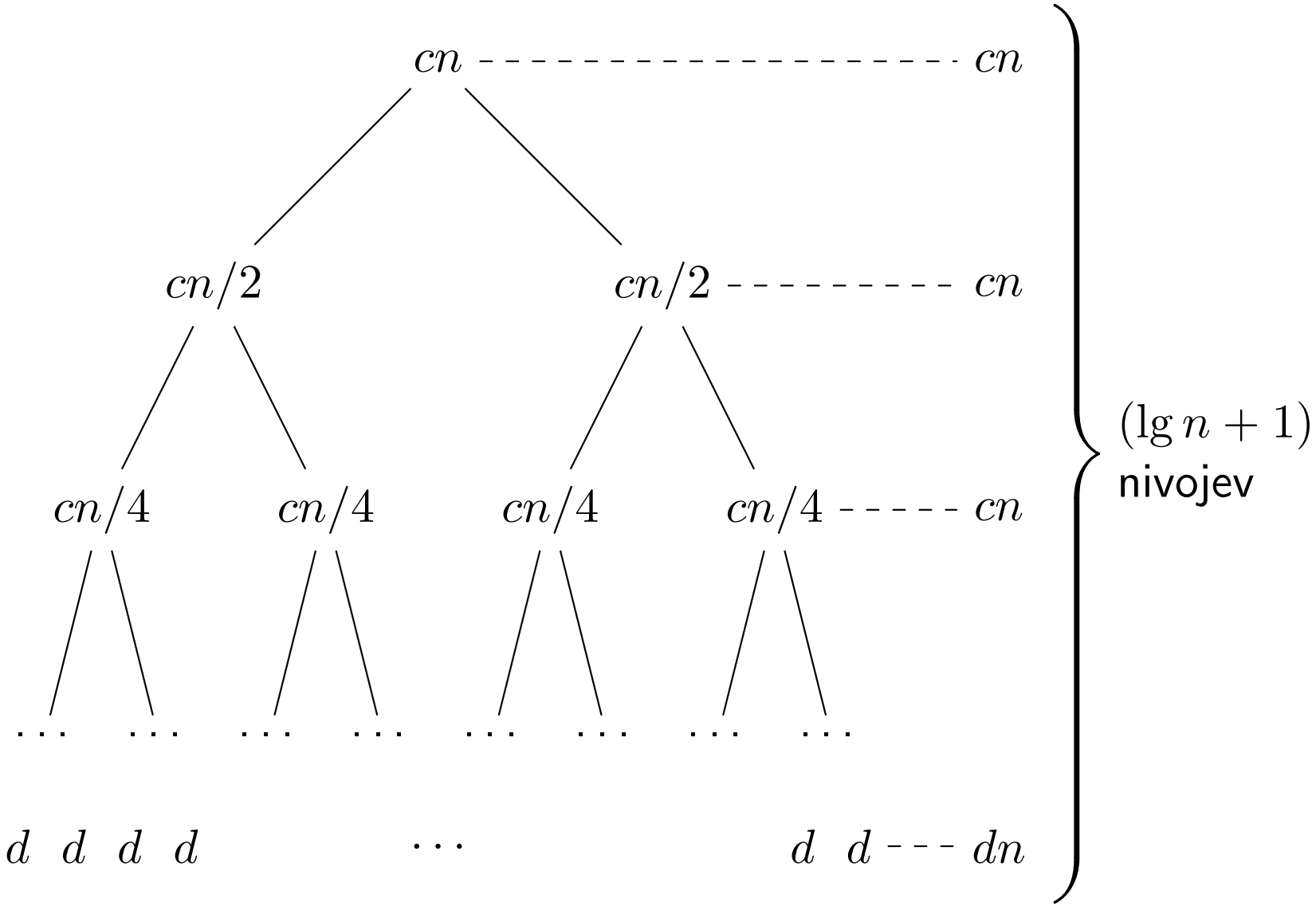
Analiza urejanja z zlivanjem



Analiza urejanja z zlivanjem



Analiza urejanja z zlivanjem



Analiza urejanja z zlivanjem

- $\lg n$ nivojev s seštevkom cn
- En nivo s seštevkom dn
- Skupaj:

$$T(n) = cn \lg n + dn = \Theta(n \log n)$$

- Ta časovna zahtevnost velja v vseh primerih (najboljšem, najslabšem in povprečnem)

Analiza algoritma bogosort

function BOGOSORT(A)
    while A ni urejena **do**
        naključno premešaj elemente A

- V najboljšem primeru (tabela je že urejena)
  - zgolj preverimo urejenost
  - $\Theta(n)$
- V najslabšem primeru
  - se ne zaključi
  - (verjetnost za kaj takega je enaka 0)
- V povprečnem primeru
  - $\Theta(n \cdot n!)$