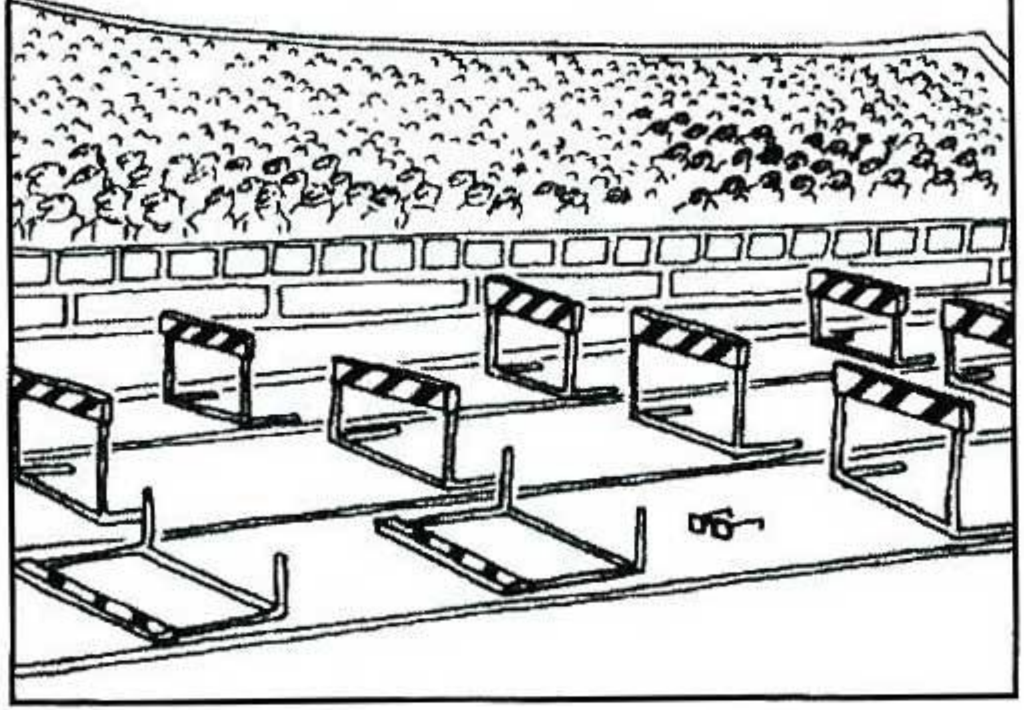
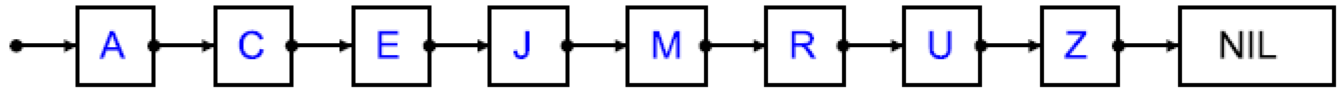


Preskočni seznam

Povezani seznam

- Primer urejenega povezanega seznama:



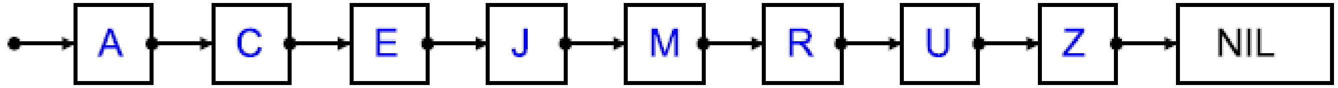
- Osnovne operacije:
  - iskanje,
  - vstavljanje,
  - brisanje.
- Časovna zahtevnost operacij?

Ali znamo bolje od O(n)?

- Iskanje v urejenem polju:
  - iskanje O(log n),
  - težave z brisanjem/vstavljanjem.
- Iskalna drevesa (binarno iskalno drevo, rdeče-črno drevo, ...)
  - spoznali jih boste v okviru tega predmeta,
  - iskanje, brisanje, vstavljanje v povprečju O(log n).
- Preskočni seznam:
  - iskanje, brisanje, vstavljanje v povprečju O(log n),
  - preprosta implementacija.

Preskočni seznam

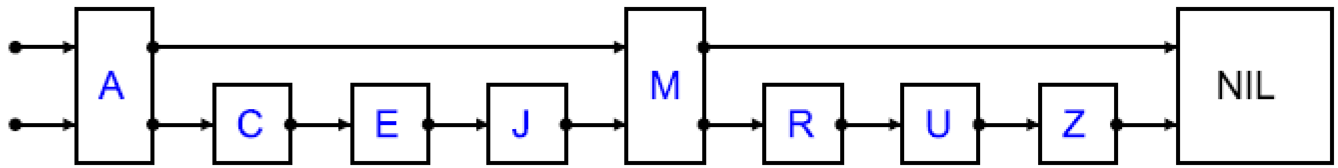
- Kako nadgraditi povezani seznam, da bo iskanje hitrejše?



- Namig...

Uvedemo “ekspresni” nivo

- Primer:

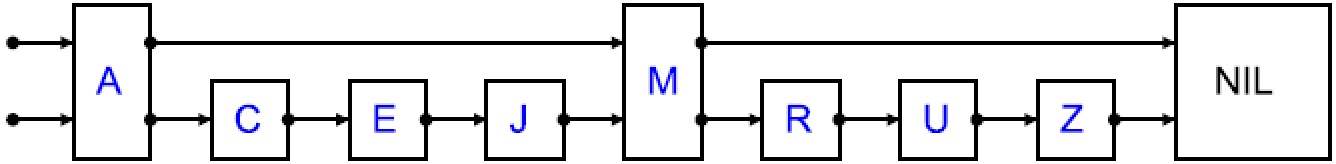


- Kako poteka iskanje?
- Izgleda obetavno, ampak:
  – Koliko elementov in
  – katere elemente povišati na ekspresni nivo?

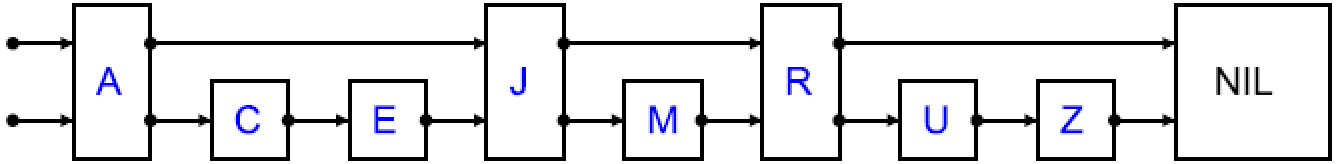
Odgovorimo najprej na drugo vprašanje...

Kako izbrati elemente za 2. nivo?

- Če lahko povišamo 2 elementa?



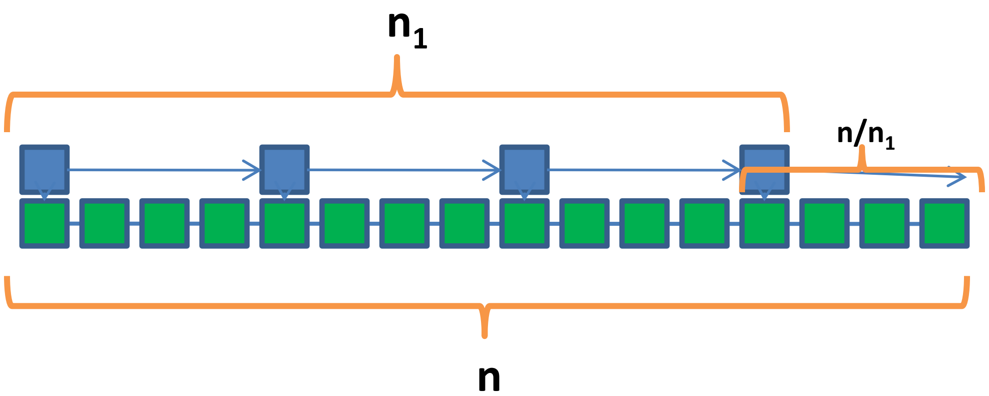
- 3 elemente?



- Zaključek: Čim bolj enakomerno!

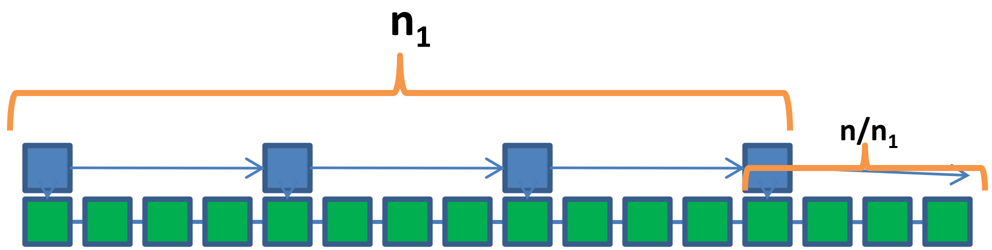
Koliko elementov?

- Poskusimo v splošnem:
  - n elementov v seznamu in
  - n_1 elementov na dodatnem nivoju,
  - $0 < n_1 <= n/2$ (zakaj?).



Kolikšen naj bo n_1, da bo število korakov pri iskanju minimalno?

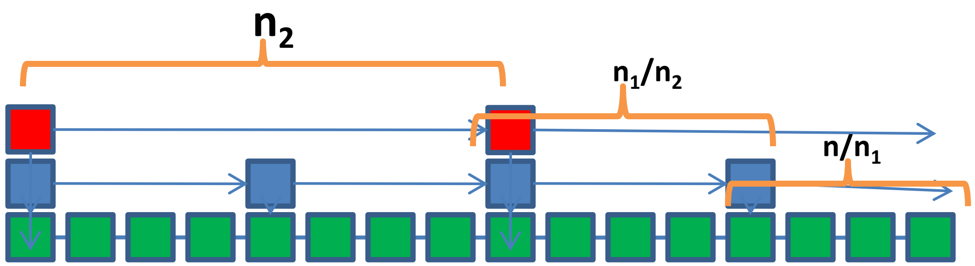
Koliko elementov?



- Število korakov = $n_1 + n/n_1$.
- Minimizirajmo!
  - Rešitev: $n_1 = \frac{n}{n_1} = \sqrt{n}$

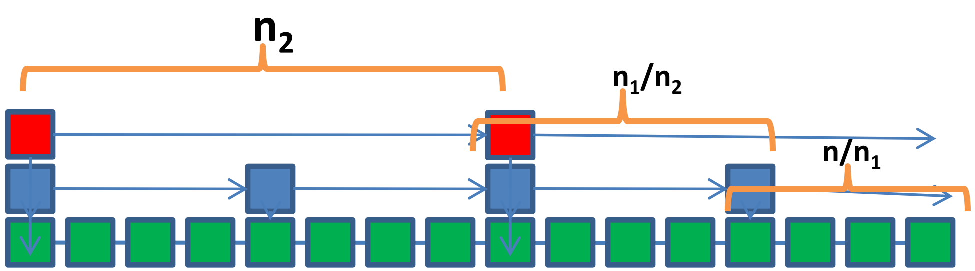
Trije nivoji

- Pri enem ekspresnem nivoju postavimo $n^{1/2}$ elementov:
  - Dosežemo čas iskanja reda velikosti $2*n^{1/2}$!
- Lahko še izboljšamo, če postavimo še en (super)ekspresni nivo?



Kolikšna naj bosta n_1 in n_2, da bo število korakov pri iskanju minimalno?

Trije nivoji

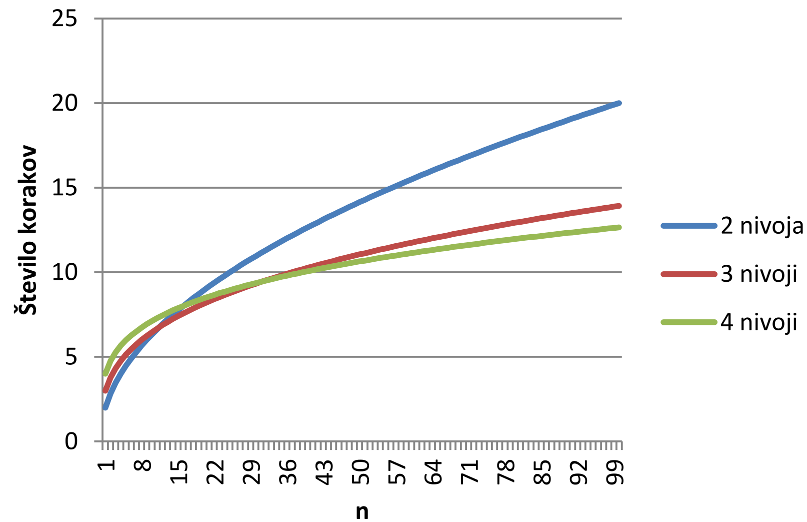


- Število korakov = $n_2 + n_1/n_2 + n/n_1$.
- Minimizirajmo!
  - Rešitev: $n_2 = \frac{n_1}{n_2} = \frac{n}{n_1} = \sqrt[3]{n}$

Koliko nivojev?

- Povečevanje števila nivojev izboljša iskanje!
- Povzemimo:

Število nivojev	Število korakov
1	$n = 1\sqrt[1]{n}$
2	$2\sqrt[2]{n}$
3	$3\sqrt[3]{n}$
k	?



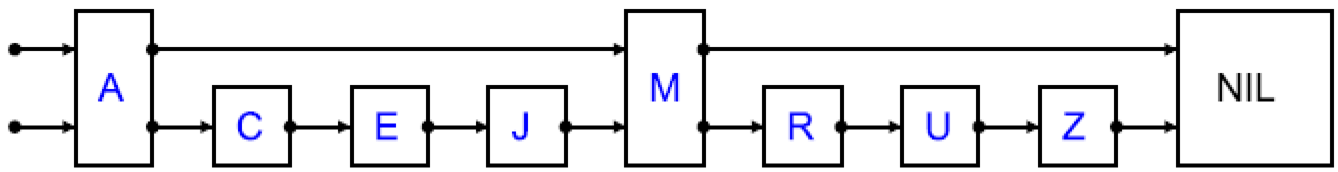
- Kateri k izbrati?
  - Spomnimo se: $0 < n_1 <= n/2$, $0 < n_2 <= n_1/2, \ldots$
  - Kolikokrat lahko razpolovimo n, da bo še večji od 1?

Število nivojev k = $\log_2 n$

- Število korakov = $\log_2 n \sqrt[\log_2 n]{n} = ?$
- Red velikosti log n
- Razmerje med nivoji je 2.
- Kako izvesti vstavljanje/brisanje, da:
  - bosta operaciji O(log n) in
  - ohranjali želene lastnosti (v povprečju):
    - na višjem nivoju pol manj elementov kot na prejšnjem,
    - enakomerno razporejeni elementi.

Vstavljanje/brisanje

- Vstavimo npr. “S”.



- Postopek vstavljanja:
  - Najprej poiščemo, kam ga moramo vstaviti.
  - Vsakega moramo vstaviti v osnovni seznam.
  - Le polovico v 2. nivo, četrtino v 3. nivo, ipd... Kako to zagotoviti?

Vstavljanje

- Vstavljanje elementa:
  - Poiščemo element, pri čemer si za vsak nivo zapomnimo zadnji obiskani element.
  - Vstavimo element v prvi nivo.
  - Ponavljamo (dokler ne pade cifra ali pridemo do najvišjega dovoljenega nivoja):
    - Vržemo kovanec.
    - Gremo nivo višje.
    - Vstavimo element v trenutni nivo.
- O(log n), da ga najdemo, O(log n), da ga vstavimo.

Brisanje

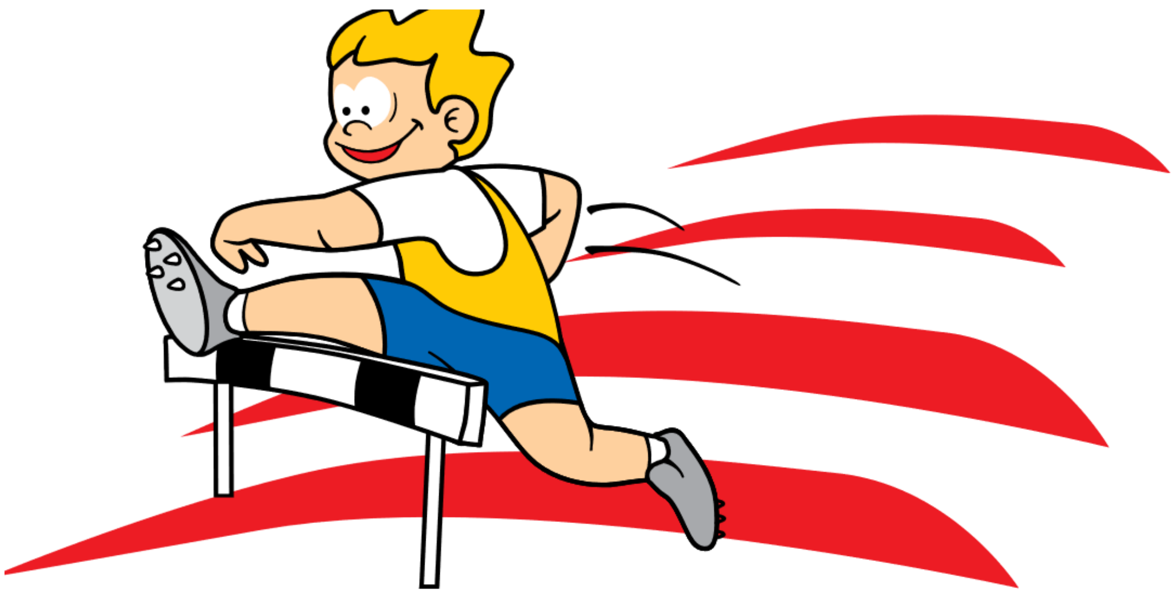
- Podobno kot pri povezanemu seznamu:
  - poiščemo element in
  - ga izbrišemo na vseh nivojih, na katerih se nahaja.
- O(log n), da ga najdemo.
- O(log n), da ga zbrišemo.
- Ker je število nivojev elementa naključno (v povprečju) ne bomo pokvarili strukture.

Povzetek

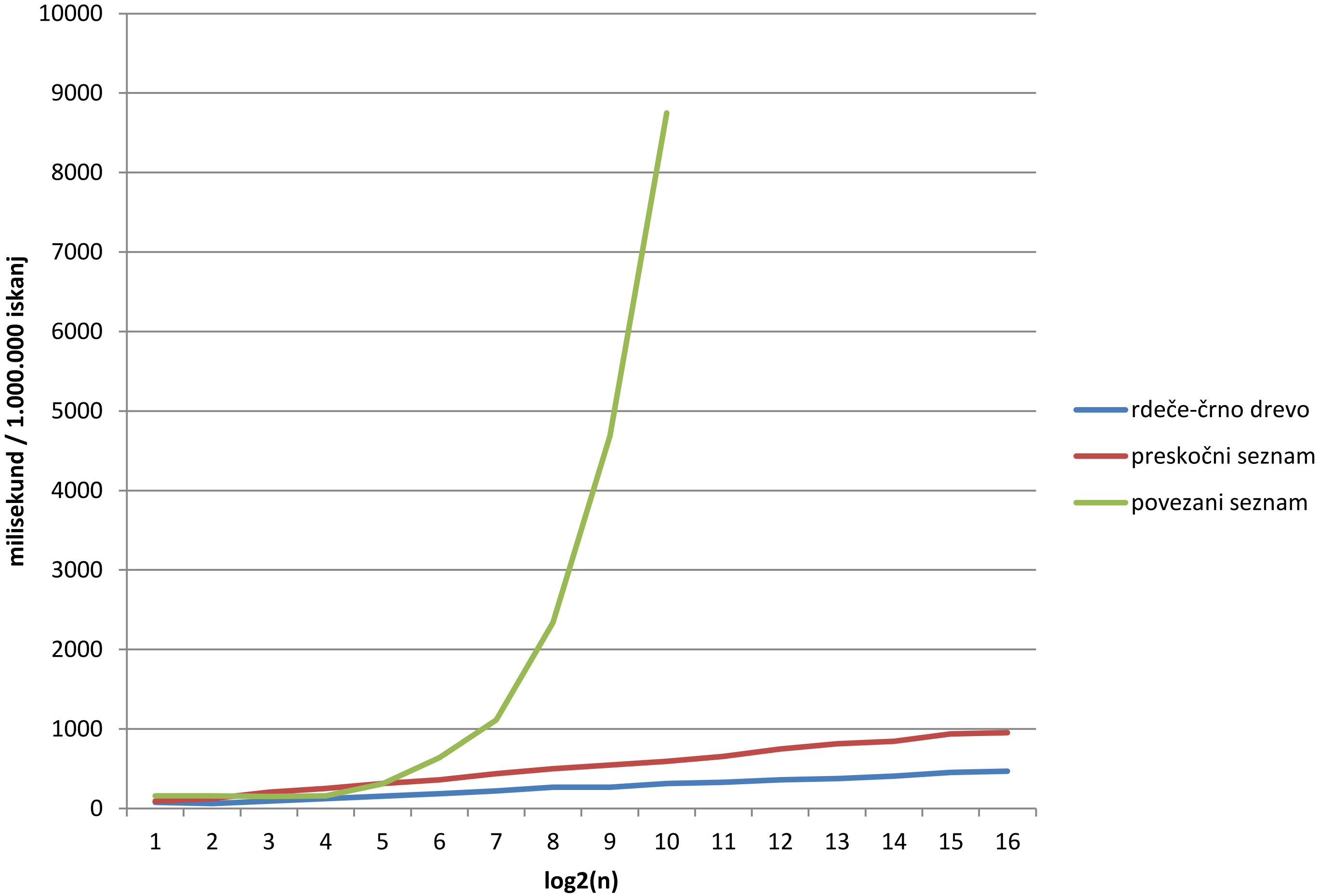
Preskočni seznam:

- Je razširitev povezanega seznama:
  - namesto enega kazalca ima vsak element polje kazalcev,
  - iskanje, vstavljanje, brisanje podobno, le na več nivojih,
  - število nivojev novega elementa je naključno.
- Časovna zahtevnost operacij O(log n).
- Preprosta implementacija.

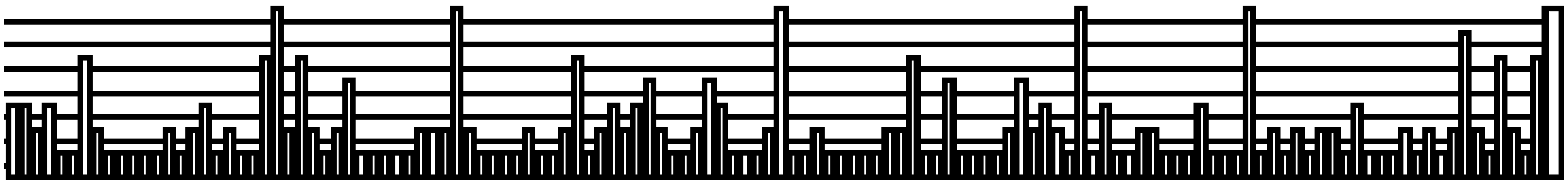
Eksperimenti

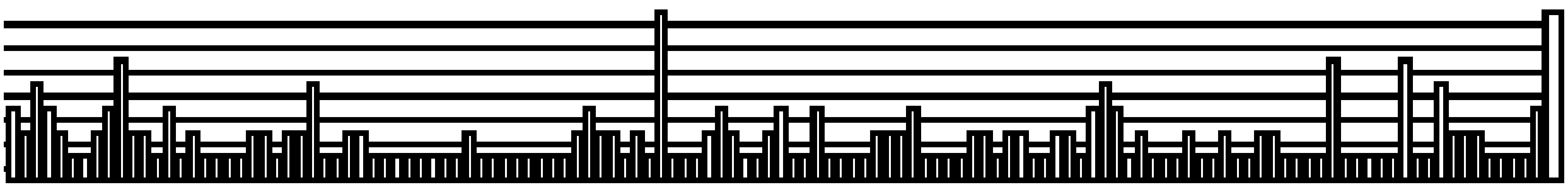


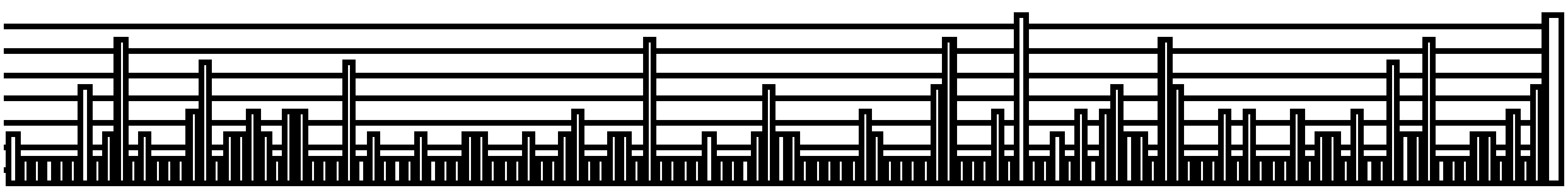
Hitrost iskanja (primerjava)



“Naključnost” preskočnih seznamov







Stabilnost hitrosti iskanja