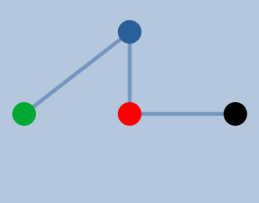
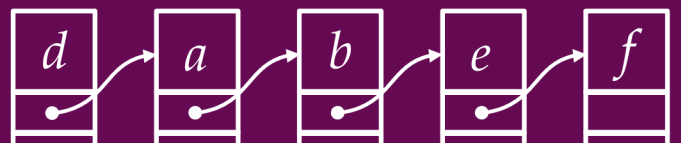
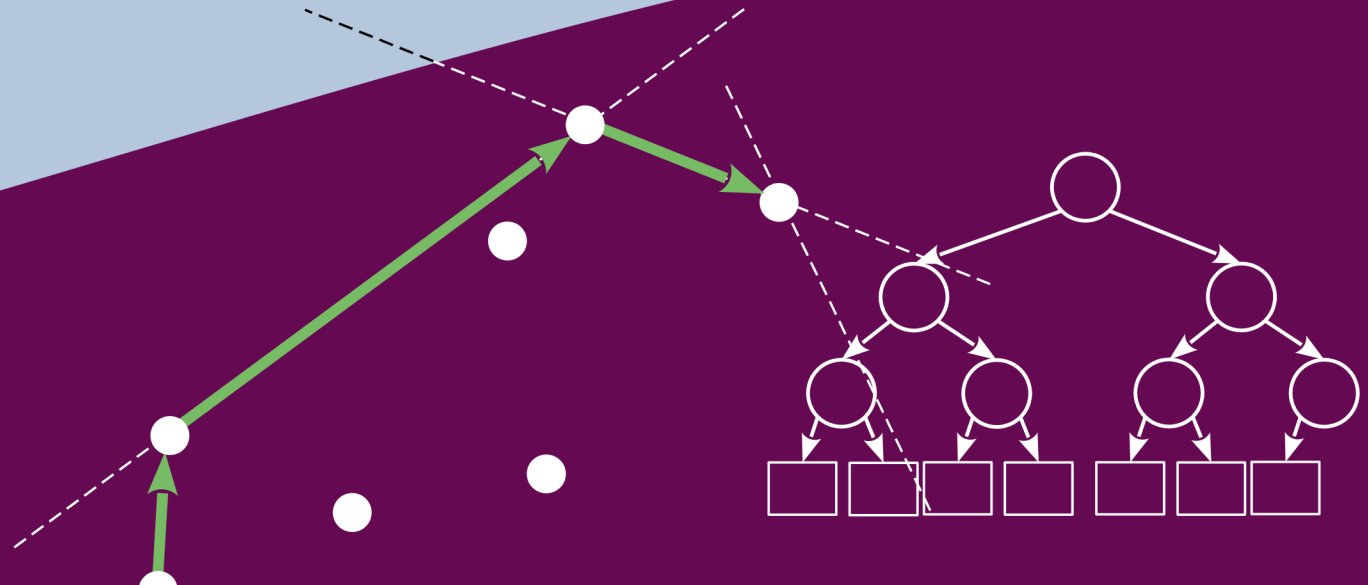
Advanced algorithms and structures data

Week 8: Random algorithms

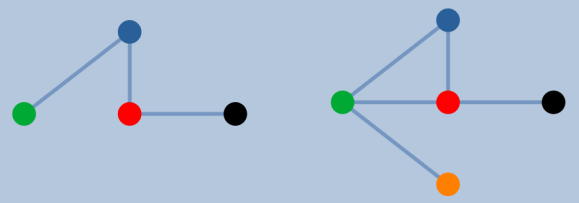


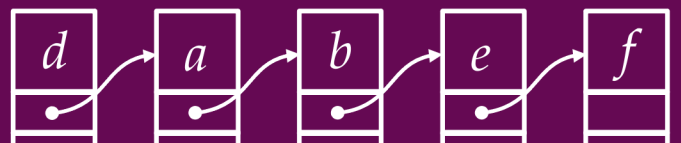


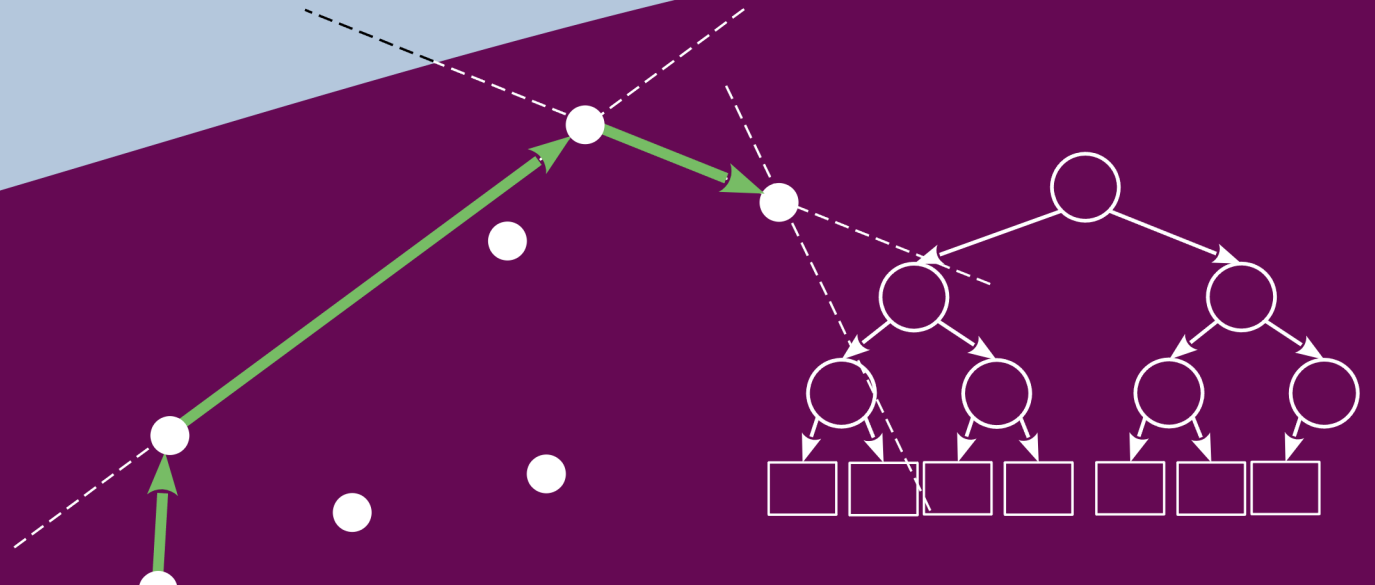


Napredni algoritmi i strukture podataka

Tjedan 8: Nasumični algoritmi





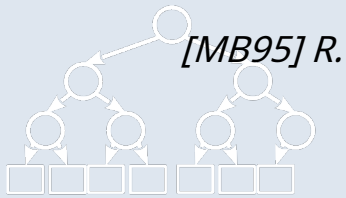


Random algorithms

- The basics
- PRNG
- Random quicksort
- Skip-lists

Lecture based on:

[MB95] R. Motwani, P. Raghavan, "Randomized algorithms", 1995, **preface and chapter 1**

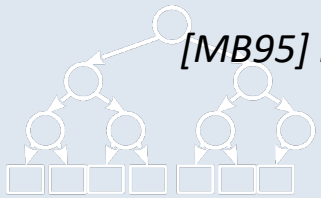


Nasumični algoritmi

- Osnove
- PRNG
- Nasumični quicksort
- Skip-liste

Predavanje bazirano na :

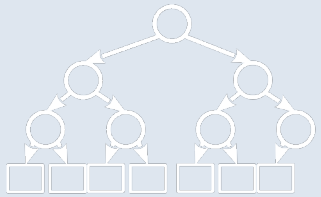
[MB95] R. Motwani, P.Raghavan „„Randomized algorithms", 1995; **predgovor i poglavlje 1**



Coincidence?

- Need?
  - Open question!
  - Rival games (Nash,...)

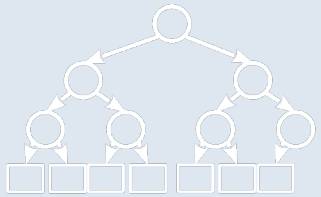
- Resource?
  - Can save other resources (time, memory)



Slučajnost?

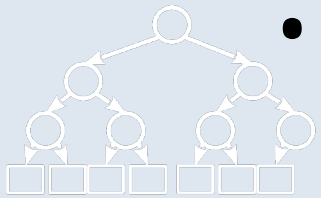
- Potreba?
  - Otvoreno pitanje!
  - Suparničke igre (Nash,...)

- Resurs?
  - Može sačuvati druge resurse (vrijeme, memorija)



Motivation

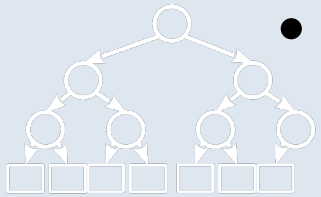
- Rival games
  - Denial-of-service attacks
    - Attacks based on computational complexity
  - Adversarial Machine Learning
  - Theft of confidential information
    - Cryptography
- Easier to better algorithms!
  - Simplicity and/or speed



Motivacija

- Suparničke igre
  - Denial-of-service napadi
    - Napadi temeljeni na računalnoj složenosti
  - Suparničko strojno učenje
  - Krađa povjerljivih podataka
    - Kriptografija

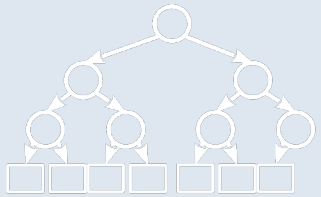
- Lakše do boljih algoritama!
  - Jednostavnost i/ili brzina



Motivation – rival games

- Attacks based on computational complexity
  - For example quicksort

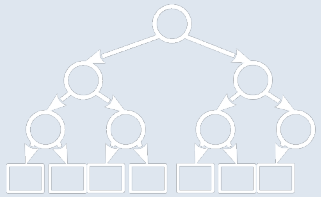
- Adversarial Machine Learning
  - Randomization matters: How to defend against strong adversarial attacks , ICML 2020.
  - On the robustness of randomized classifiers to adversarial examples , Arxiv 2021.



Motivacija – suparničke igre

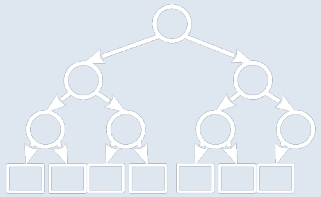
- Napadi temeljeni na računalnoj složenosti
  - Npr. quicksort

- Suparničko strojno učenje
  - Randomization matters: How to defend against strong adversarial attacks, ICML 2020.
  - On the robustness of randomized classifiers to adversarial examples, Arxiv 2021.



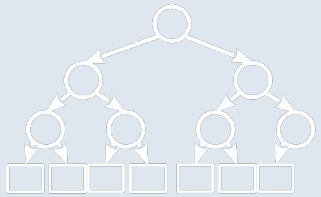
Motivation – better algorithms

- Pregnancy testing
  - The fastest deterministic algorithm - ( )
  - The fastest probabilistic algorithm - (")



Motivacija – bolji algoritmi

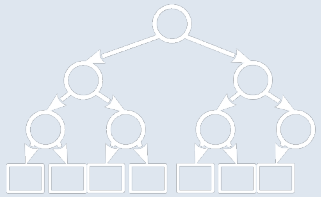
- Testiranje primalnosti
  - Najbrži deterministički algoritam – $O(n^5)$
  - Najbrži probabilistički algoritam - $O(n^3)$



Motivation – a paradigm of algorithm design

1. Development of an efficient probabilistic algorithm
2. Derandomization (complex!)
3. Obtained efficient deterministic algorithm

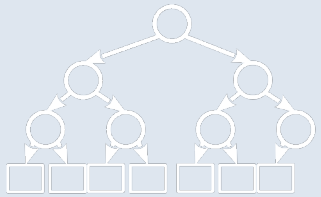
- Examples
  - Primality test in polynomial time
    - probabilistic algorithm (2003) ->deterministic algorithm (2004)
  - An undirected graph connectivity test in logarithmic space
    - probabilistic algorithm (1979) ->deterministic algorithm (2008)



Motivacija – paradigma dizajna algoritama

1. Izrada efikasnog probabilističkog algoritma
2. Derandomizacija (složeno!)
3. Dobiven efikasni deterministički algoritam

- Primjeri
  - Test primalnosti u polinomijalnom vremenu
    - probabilistički algoritam (2003) -> deterministički algoritam (2004)
  - Test povezanosti neusmjerenog grafa u logaritamskom prostoru
    - probabilistički algoritam (1979) -> deterministički algoritam (2008)

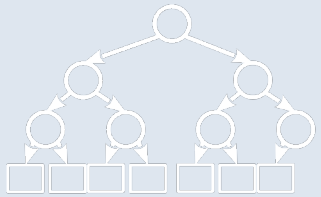


Random algorithms

- Algorithms with access to a source of independent, unbiased random bits
  - Random bits affect calculations

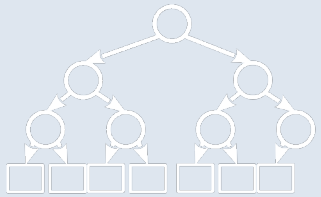
- Algorithms with theoretical guarantees!

- Stochastic algorithms are not the topic of this lecture
  - They have no solid theoretical guarantees, "only" empirical results
  - Optimization algorithms: such as evolutionary, tabu-search, simulated annealing...



Nasumični algoritmi

- Algoritmi sa pristupom izvoru nezavisnih, nepristranih slučajnih bitova
  - Slučajni bitovi utječu na izračune
- Algoritmi sa teorijskim jamstvima!
- Stohastički algoritmi – nisu tema ovog predavanja
  - Nemaju čvrstih teorijskih jamstava, „samo” empirijski rezultati
  - Optimizacijski algoritmi: poput evolucijskih, tabu-pretraživanja, simuliranog kaljenja...

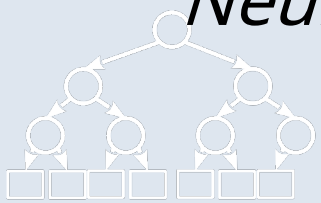


Random algorithms

- Algorithms with access to a source of independent, unbiased random bits
  - Random bits affect calculations

- Pseudo-random numbers

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."*John von Neumann*

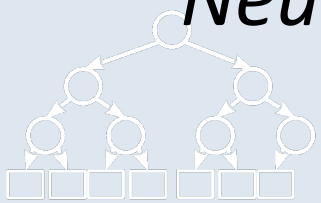


Nasumični algoritmi

- Algoritmi sa pristupom izvoru nezavisnih, nepristranih slučajnih bitova
  - Slučajni bitovi utječu na izračune

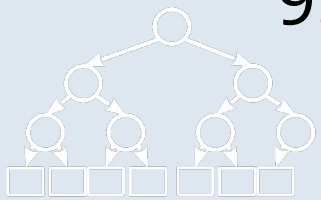
- Pseudo-slučajni brojevi

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.„ *John von Neumann*



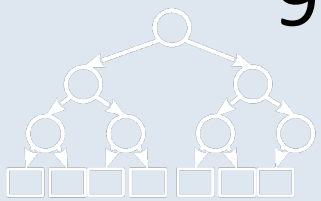
Random algorithms - paradigms

1. Deception of the opponent
2. Random sampling -> eg from the population
3. Finding "witnesses"-> eg tests for evidence
4. Fingerprinting and hashing
5. Random redeployment
6. Load balancing -> distributed computing
7. Fast-mixing Markov chains -> approximate counts
8. Isolation and breaking of symmetries -> coordination
9. Probabilistic methods and existence proofs -> p>0



Nasumični algoritmi - paradigme

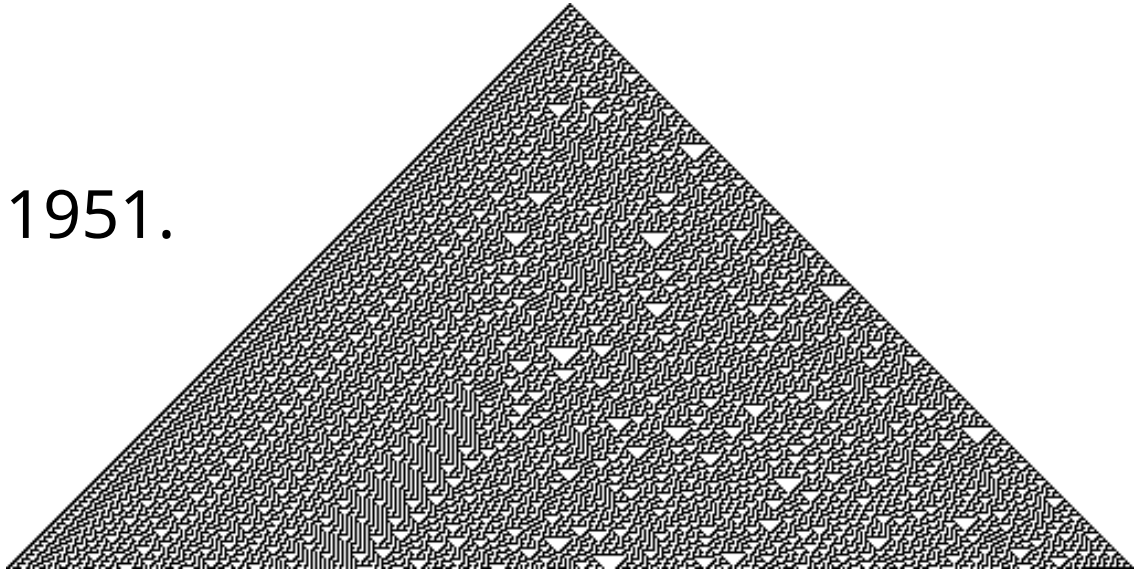
1. Prevara protivnika
2. Slučajno uzorkovanje -> npr. iz populacije
3. Pronalazak „svjedoka”-> npr. testovi za dokaze
4. Fingerprinting i hashing
5. Slučajno preraspoređivanje
6. Balansiranje opterećenja -> raspodijeljeno računarstvo
7. Brzo-miješajući Markovljevi lanci -> približna brojanja
8. Izolacija i razbijanje simetrija -> koordinacija
9. Probabilističke metode i dokazi postojanja -> $p>0$



Pseudo-random number generators

- Rich development

- Linear congruent generators (LCG) – 1951.

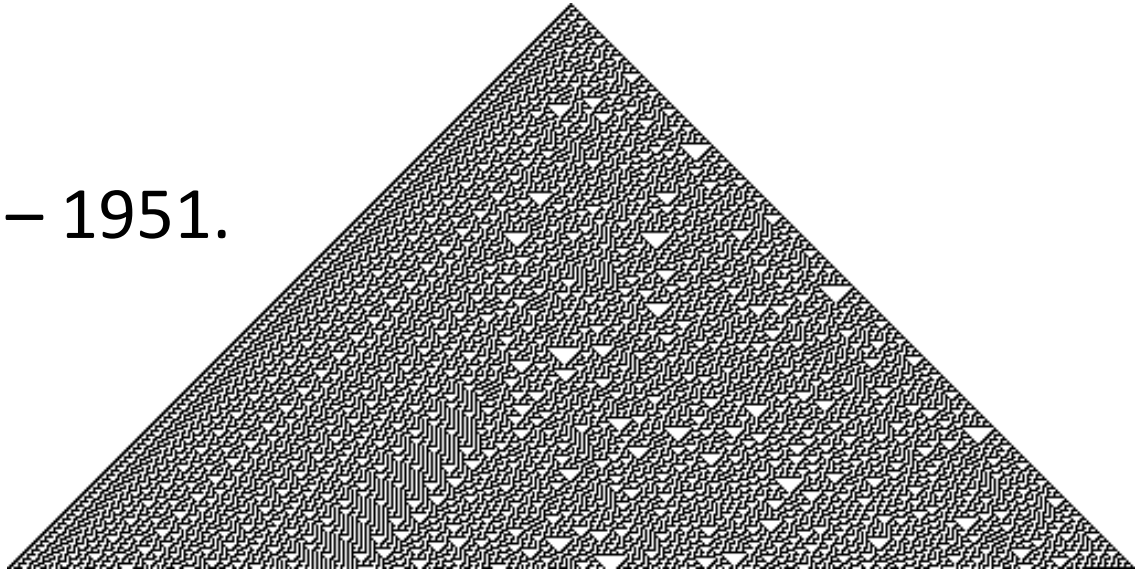


- Rule 30 – 1983

- Mean square and Weyl sequence generator – 2017

Generatori pseudo-slučajnih brojeva

- Bogat razvoj
- Linearni kongruentni generatori (LCG) – 1951.
- Pravilo 30 – 1983.
- Generator srednjeg kvadrata i Weylovog slijeda – 2017.

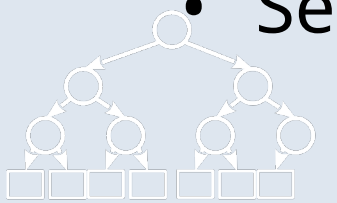


Linear congruent generator

- One of the simplest
- Widely used (C, Java)

$$!"\#= \quad (!+ \quad)$$

- Constants: *a, c, m*
- "Seed”: $

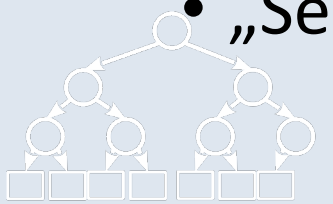


Linearni kongruentni generator

- Jedan od najjednostavnijih
- Široko korišten (C, Java)

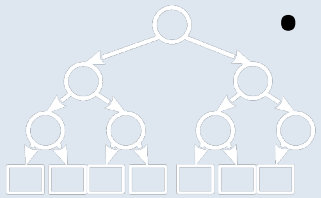
$$X_{n+1} = (aX_n + c)\ mod\ m$$

- Konstante: a,c,m
- „Seed”: X_0



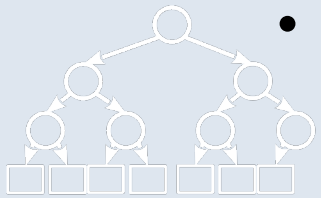
Random algorithms - types

- Las Vegas
  - It always gives the correct answer or failure info
  - Run time varies
- Monte Carlo
  - Running time limited
  - It never returns the correct answer
    - Failure
    - Incorrect answer
  - Independent successive starts -> arbitrary reduction of chance of failure



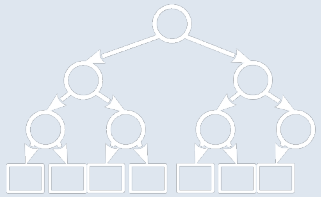
Nasumični algoritmi - vrste

- Las Vegas
  - Uvijek daje točan odgovor ili info o neuspjehu
  - Vrijeme izvođenja varira
- Monte Carlo
  - Vrijeme izvođenja ograničeno
  - Nekad ne vraća točan odgovor
    - Neuspjeh
    - Netočan odgovor
  - Nezavisna uzastopna pokretanja -> proizvoljno smanjenje šanse neuspjeha



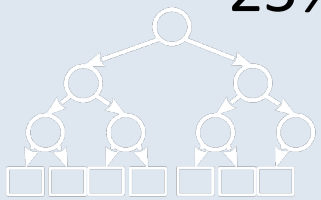
Complexity classes for decision problems

- **P**–all of which can be solved in polynomial time
- **NP**–all of which the verification of the solution can be done in polynomial time
- **ZPP**(zero-error probabilistic polynomial) – all that have Las Vegas algorithm with expected polynomial duration
- **RP**(randomized polynomial) – all of which have a Monte Carlo algorithm with a one-sided error and a polynomial duration in the worst case
- **PP**(probabilistic polynomial) – all that have a Monte Carlo algorithm with a two-sided error (no worse than 50%) and a polynomial duration in the worst case
- **BPP**(bounded-error probabilistic polynomial) – PP, but error no worse than 25%



Klase složenosti za probleme odlučivanja

- **P** – svi koji mogu biti riješeni u polinomijalnom vremenu
- **NP** – svi za koje verifikacija rješenja može biti obavljena u polinomijalnom vremenu
- **ZPP** (zero-error probabilistic polynomial) – svi koji imaju Las Vegas algoritam sa očekivanim polinomijalnim trajanjem
- **RP** (randomized polynomial)– svi koji imaju Monte Carlo algoritam sa jednostranom greškom i polinomijalno trajanje u najgorem slučaju
- **PP** (probabilistic polynomial)– svi koji imaju Monte Carlo algoritam sa dvostranom greškom (ne gore od 50%) i polinomijalno trajanje u najgorem slučaju
- **BPP** (bounded-error probabilistic polynomial) – PP, ali greška ne gora od 25%

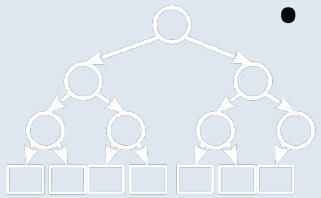


Random quicksort

- Quicksort
  - best case ($ log)
  - worst case (!)

```
Quicksort(lo,hi,A):
1. if lo >= 0 && hi >= 0 && lo < hi then
2.         p := partition(A, lo, hi) // pivot selection, splitting
3.         quicksort(A, lo, p) // Note: the pivot is now included
4.         quicksort(A, p + 1, hi)
```

- deterministic fjapartition
  - worst case (%) –open to **complexity-based attacks**

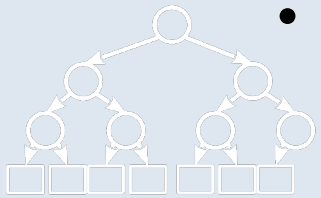


Nasumični quicksort

- Quicksort
  - najbolji slučaj $O(n \cdot \log n)$
  - najgori slučaj $O(n^2)$

```
Quicksort(lo,hi,A):
1. if lo >= 0 && hi >= 0 && lo < hi then
2.     p := partition(A, lo, hi) // pivot selection, splitting
3.     quicksort(A, lo, p) // Note: the pivot is now included
4.     quicksort(A, p + 1, hi)
```

- deterministička fja `partition`
  - najgori slučaj $O(n^2)$ – otvoreno za **napade bazirane na složenosti**

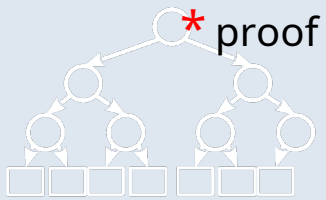


Random quicksort

- random fjapartition
  - Uniformly random selection of pivots from a given range
  - worst case (%) –vanishing probability
  - In anticipation (1 log)*
    - for each input
    - even against the "enemy"
  - Runtime random,even for repeated entry

- Decoupling the structure of the input from the functioning of the algorithm

* proof in [MB95]

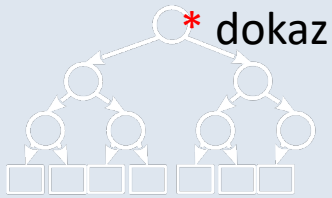


Nasumični quicksort

- nasumična fja `partition`
  - Uniformno slučajan odabir pivota iz danog raspona
  - najgori slučaj $O(n^2)$ – iščezavajuća vjerojatnost
  - U očekivanju $O(n \cdot \log n)$ *
    - za svaki ulaz
    - čak i protiv „neprijatelja”
  - Vrijeme izvođenja slučajno, <u>čak i za ponovljeni ulaz</u>

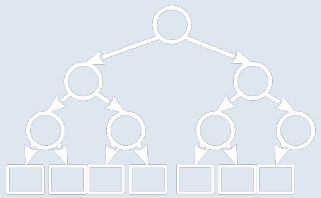
- Odvajanje (decoupling) strukture ulaza od funkcioniranja algoritma

* dokaz u [MB95]



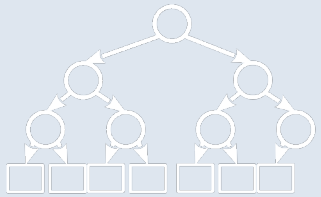
Skip lists

- *Pugh, William: "Skip lists: a probabilistic alternative to balanced trees", Communications of the ACM 33, June 1990, pp. 668-676*
- **Basic lack of leaf**:O(*n*) search
- **Limiting property of trees**: by nature a hierarchical structure, logically unsuitable for all applications.
- Skip lists:
  - key operations $O(\log_2 n)$...O(n)
  - **there is no hierarchy**
  - relatively simple programming
  - Parallel access!!!



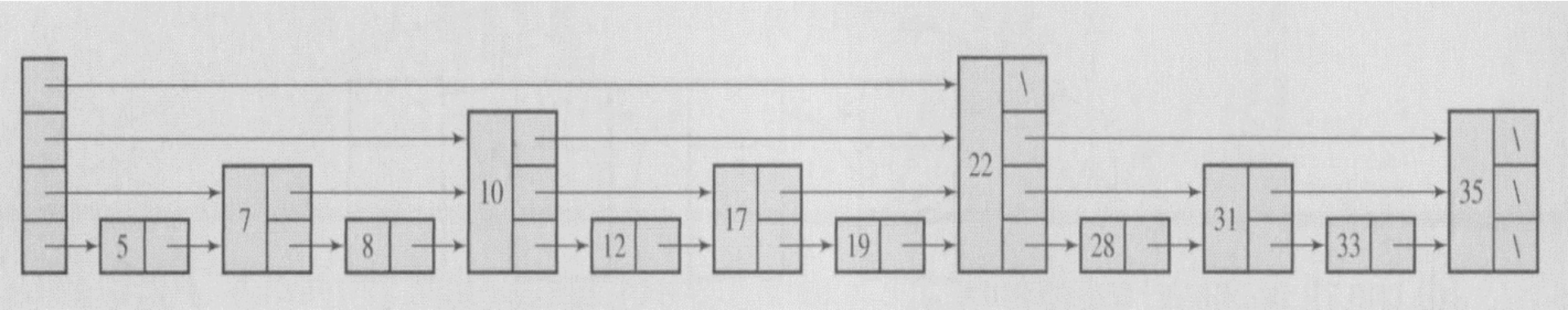
Preskočne liste

- *Pugh, William: "Skip lists: a probabilistic alternative to balanced trees“, Communications of the ACM 33, June 1990, pp. 668-676*
- **Osnovni nedostatak lista**: O(*n*) pretraživanje
- **Ograničavajuće svojstvo stabala**: po prirodi hijerarhijske strukture, logički neprikladne za sve primjene.
- Skip liste:
  - ključne operacije $O(\log_2 n)...O(n)$
  - **nema hijerarhije**
  - relativno jednostavno programiranje
  - Paralelni pristup!!!



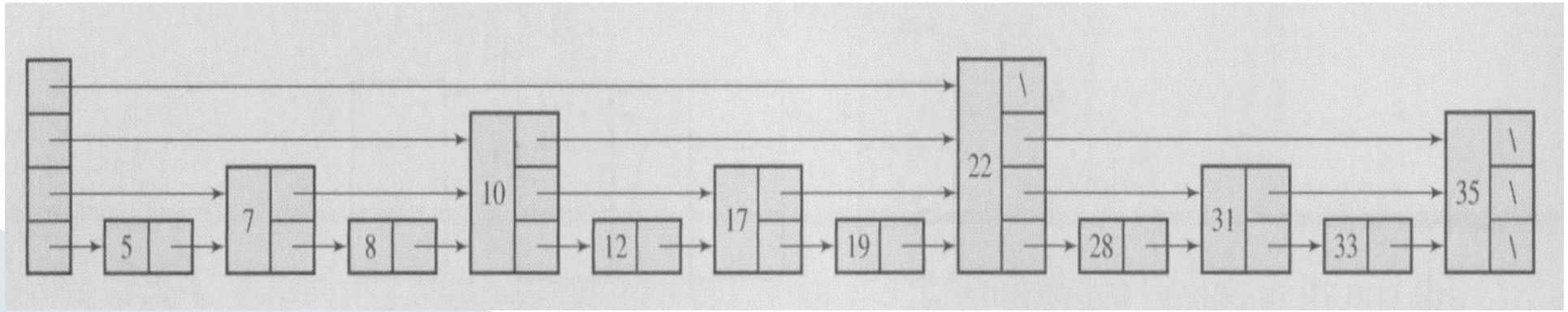
Skip lists - an extract of ideas from trees

- Node level = number of pointers
  - In the example below, head degree = 4
- Maintaining this perfect structure
  - Complicated and inefficient - "acting" tree
  - Restructuring of all nodes behind the change



Preskočne liste – izvod ideje iz stabala

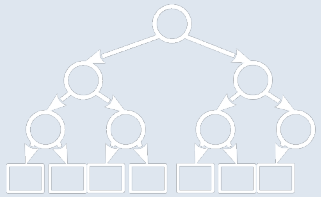
- Stupanj (level) čvora = broj pokazivača
  - U primjeru niže, stupanj glave = 4
- Održavanje ovakve savršene strukture
  - Komplicirano i neučinkovito – „glumi” stablo
  - Restrukturiranje svih čvorova iza promjene



Skip lists

- **Withdrawal**from the request forproper hierarchical arrangement of nodes

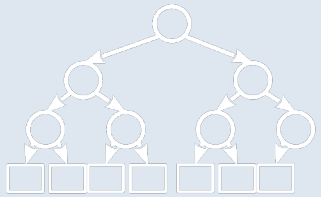
- **strives**is likely to be achievedcorrect*distribution*their degrees.
  - The histogram of the number of nodes per degree should tend to the histogram of the ideal case



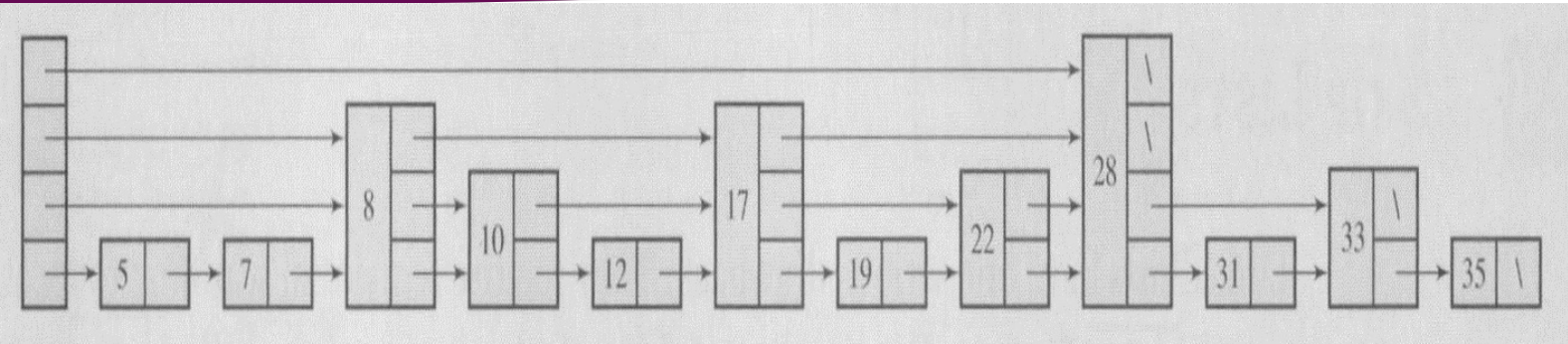
Preskočne liste

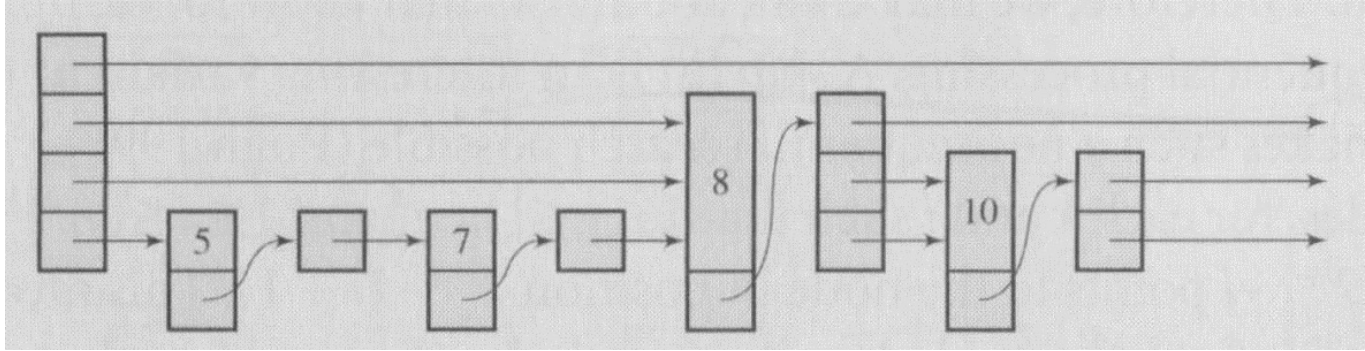
- **Odustajanje** od zahtjeva za pravilnim hijerarhijskim rasporedom čvorova

- **nastoji** se vjerojatnosno postići pravilna *razdioba* njihovih stupnjeva.
  - Histogram broja čvorova po stupnjevima bi trebao težiti histogramu idealnog slučaja



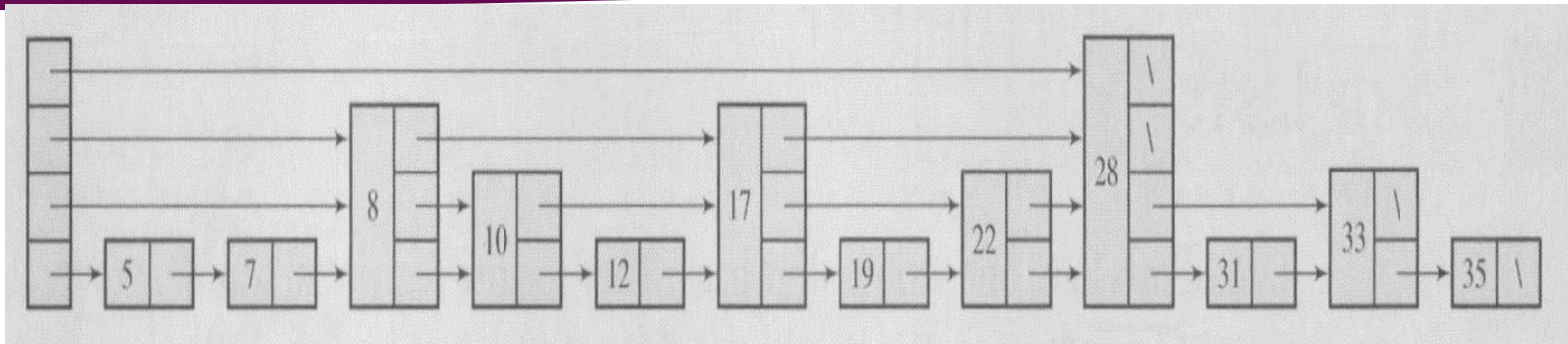
Skip lists

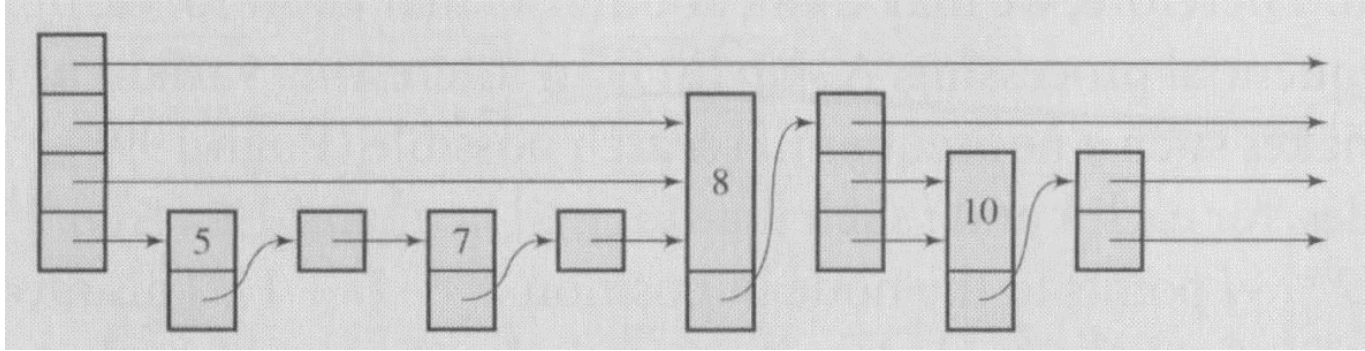




- Data access speed is u**average**comparable to the speed in the AVL or RB tree
- Worse guarantees for the worst case
- Relationship between trees and skip list

Preskočne liste

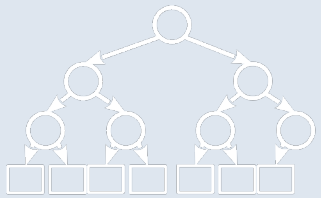




- Brzina pristupa podatcima je u **prosjeku** sumjerljiva brzini u AVL ili RB stablu
- Lošija jamstva za najgori slučaj
- Odnos između stabala i preskočne liste

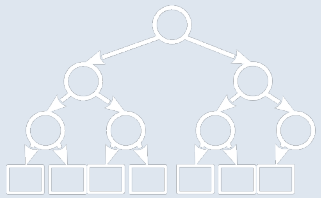
Skip lists

- The usable structure of the skip list depends on two factors:
  1. the intended capacity n
     - assumed maximum number of elements in the list
  2. probabilities of individual node degrees
     - determines the distribution
     - most often only probability is selected p of moving the node to a higher level
     - defines **geometric distribution**
     - transitions to a higher level are repeated until the first failure



Preskočne liste

- Uporabna struktura skip liste ovisi o dva čimbenika:
  1. predviđenom kapacitetu *n*
     - pretpostavljeni najveći broj elemenata u listi
  2. vjerojatnosti pojedinih stupnjeva čvora
     - određuje razdiobu
     - najčešće se odabire samo vjerojatnost *p* prijelaska čvora u višu razinu
     - definira **geometrijsku razdiobu**
     - prijelasci u višu razinu ponavljaju se sve do prvog neuspjeha



Skip lists

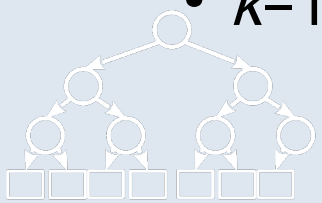
- **Capacity** n and **probability** p determine all the theoretical features of the skip list

- The probability $P(k)$ that the new node ultimately achieves kth degree

- $P(k) = [P(\text{transition})]^{k-1} \cdot P(\text{remaining}) = p^{k-1} \cdot (1 - p)$

Geometric distribution

- k–1 successful jumps to a higher level and one, final, failure

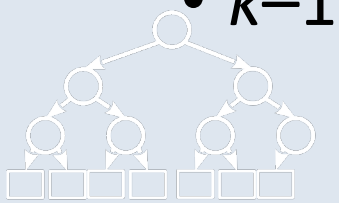


Preskočne liste

- **Kapacitet** *n* i **vjerojatnost** *p* određuju sve teorijske značajke preskočne liste
- Vjerojatnost P(*k*) da novi čvor u konačnici postigne *k*-ti stupanj
- $P(k) = [P(\text{prijelaz})]^{k-1} \cdot P(\text{ostanak}) = p^{k-1} \cdot (1 - p)$

Geometrijska razdioba

- *k*–1 uspješnih preskoka u višu razinu i jedan, konačni, neuspjeh



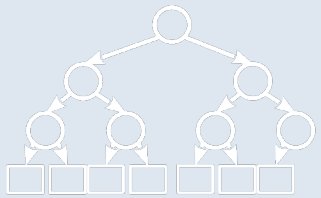
Skip lists

- Expected (“mean”) degree of nodes in the list (“mean height” of the list) predicted capacity n

$$E(k) = \sum_{k=1}^{\infty} k \times P(k) = (1-p) \sum_{k=1}^{\infty} k \times p^{k-1}$$

$$\sum_{k=1}^{\infty} k \times p^{k-1} = \frac{1}{(1-p)^2}$$

$$\Rightarrow \quad E(k) = \sum_{k=1}^{\infty} k \times P(k) = \frac{1}{1-p}$$



Preskočne liste

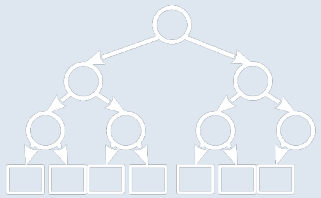
- Očekivani (“srednji”) stupanj čvorova u listi (“srednja visina” liste) predviđenom kapacitetu n

$$E(k) = \sum_{k=1}^{\infty} k \cdot P(k) = (1-p) \sum_{k=1}^{\infty} k \cdot p^{k-1}$$

$$\sum_{k=1} k \cdot p^{k-1} = \frac{1}{(1-p)^2}$$

=>

$$E(k) = \sum_{k=1}^{\infty} k \cdot P(k) = \frac{1}{1-p}$$



Skip lists

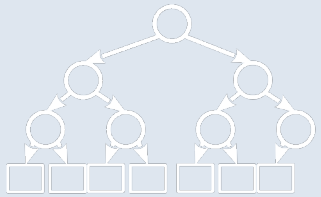
- Correct number n_k nodes k-th degree is a random variable, so it can be calculated **expectation** $E(n_k)$

- n_k with n the total number of inserted numbers in the skip list has a binomial distribution

$$n_k \sim B(n_k; n, P(k))$$

- Therefore, the expectation $E(n_k)$ is

$$E(n_k) = n \cdot P(k) = n \cdot p^{k-1} \cdot (1 - p)$$



Preskočne liste

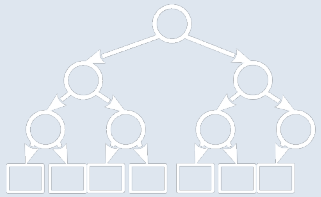
- Točan broj n_k čvorova k-tog stupnja je <u>slučajna varijabla</u> pa se može izračunati **očekivanje** $E(n_k)$

- n_k s $\boldsymbol{n}$ ukupno ubačenih brojeva u skip listu ima binomnu razdiobu

$$n_k \sim B(n_k; n, P(k))$$

- Prema tome, očekivanje $E(n_k)$ je

$$E(n_k) = n \cdot P(k) = n \cdot p^{k-1} \cdot (1 - p)$$



Jump lists - construction - number of levels

- In a perfectly constructed skip-list there will be only one node of the highest degree *h*

$$\$ \ "\#\$ \geq 1$$

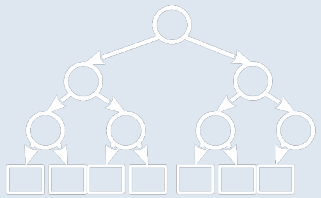
$$h \leq 1 + \ \% = 1 + \ \$^{\frac{1}{\%}}$$

  - We take

$$h = \ (1 + \ \$)^{\frac{1}{\%}}$$

Example: p=0.5, n=12

$$h = \ 1 + (\ !12 = \ 3.6 = 4 \ (\ )$$



Preskočne liste – konstruiranje – broj razina

- U savršeno konstruiranoj skip-listi bit će samo jedan čvor najvišeg stupnja *h*

$$n \cdot p^{h-1} \geq 1$$

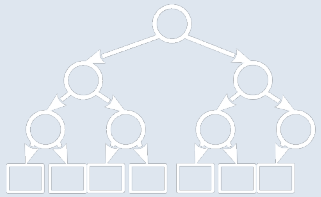
$$h \leq 1 + log_p \frac{1}{n} = 1 + log_{\frac{1}{p}} n$$

  - Uzimamo

$$h = floor(1 + log_{\frac{1}{p}} n)$$

Primjer: p=0.5, n=12

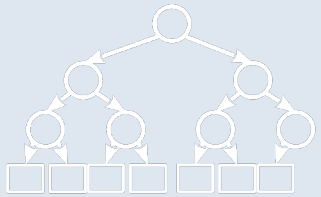
$$h = floor(1 + log_2 12) = floor(4.6) = 4$$



Jump lists - construction - sampling

1. Sampling successive climbs with an interruption at *h*
   - "Surplus" is distributed at the highest level

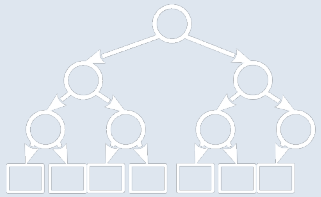
- Direct sampling - uses only one random number per insertion
  2. From the cut cumulative distribution F(k) -> break at h
     - "Surplus" is distributed at the highest level
  3. From the quantized cumulative distribution H(k)
     - Quantization - rounding
     - "excess" is distributed by histogram



Preskočne liste – konstruiranje – uzorkovanje

1. Uzastopno uzorkovanje uspona sa prekidom na *h*
   - „Višak” se rasporedi na najvišoj razini

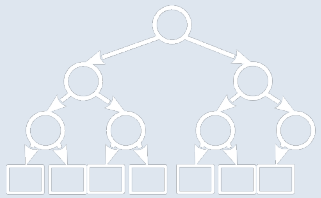
- Direktno uzorkovanje – koristi samo jedan sluč.broj po umetanju
  2. Iz odrezane kumulativne distribucije F(k) -> prekid na h
     - „Višak” se rasporedi na najvišoj razini
  3. Iz kvantizirane kumulativne distribucije H(k)
     - Kvantizacija - zaokruživanje
     - „višak” se rasporedi po histogramu



Jump lists - construction - sampling 1

- Successive sampling of the ascent with a break at h
  - "Surplus" is distributed at the highest level

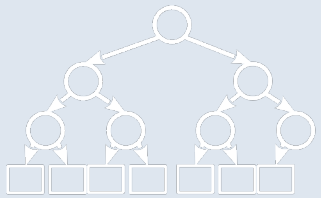
```
randomLevel()
    lvl := 1
    -- random() that returns a random value in [0...1)
    while random() < p and lvl < MaxLevel do
        lvl := lvl + 1
    return lvl
```



Preskočne liste – konstruiranje – uzorkovanje 1

- Uzastopno uzorkovanje uspona sa prekidom na h
  - „Višak” se rasporedi na najvišoj razini

```
randomLevel()
    lvl := 1
    -- random() that returns a random value in [0...1)
    while random() < p and lvl < MaxLevel do
        lvl := lvl + 1
    return lvl
```

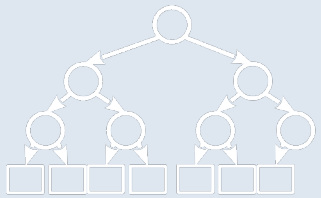


Skip lists - construction - sampling 2

- Direct sampling from the "cut" cumulative distribution F(k) (break at h)
  - "Surplus" is distributed at the highest level
  - ∉) ≤ ; (*h* = 1) ( )

    randomLevelDirect()

    1. |v| =1
    2. r=random()
    3. while r>F(|v|):
       1. |v|:=|v|+1
    4. return |v|

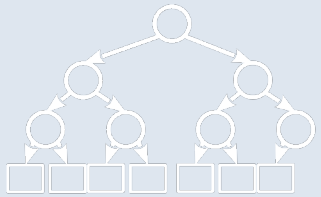


Preskočne liste – konstruiranje – uzorkovanje 2

- Direktno uzorkovanje iz „odrezane” kumulativne distribucije F(k) (prekid na h)
  - „Višak” se rasporedi na najvišoj razini
  - $F(k) = P(x \leq k); F(h) = 1$

randomLevelDirect()

1. |v| =1
2. r=random()
3. while r>F(|v|):
   1. |v|:=|v|+1
4. return |v|



Jump lists - construction - histogram

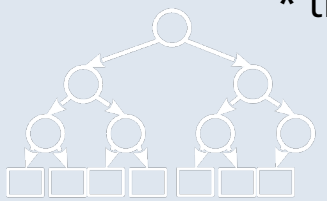
- Let's arrange the cumulative histogram as expected (previous slides)

$$(\leq) = \cdot (1 - (>)) = \cdot (1 -)$$

Number of nodes of level less than or equal to the cumulative histogram:

$$= ( ) \quad /01 \quad (\quad (\quad)) = \quad 1 \left(1 - \left(1 \quad \right)\right)$$

* trivial, (0) = 0



Preskočne liste – konstruiranje – histogram

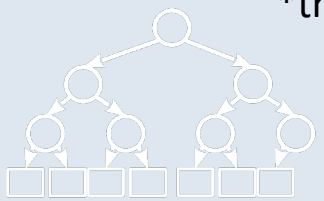
- Složimo kumulativni histogram prema očekivanju (prethodni slideovi)

$$\mathbb{E}(n_{x \leq k}) = n \cdot (1 - P(x > k)) = n \cdot (1 - p^k)$$

Broj čvorova razine manje ili jednake k u kumulativnom histogramu:

$$H(k) = ceil\left(\mathbb{E}(n_{x \leq k})\right) = ceil\left(n \cdot \left(1 - p^k\right)\right)$$

*trivijalno, $H(0) = 0$



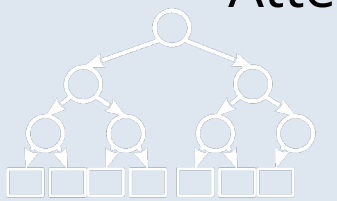
Jump lists - construction - histogram

- p=0.5, n=12, h=4

$$( ) = \quad (\quad 1 \ (1 - \ _1))$$

(1) = 12 1 ((1 − 0.5#) = 6)(2) =
12 1 (1 − 0.5%) = 9)
3 ( ) 12 1 ((1 − 0.5$_2$) = 1)0.5 = 11(4 =) 12 1
(1(−)0.5$_2$) = (11.25 = 12) ( )

- Attempting to calculate for all higher levels gives ( ) = 12



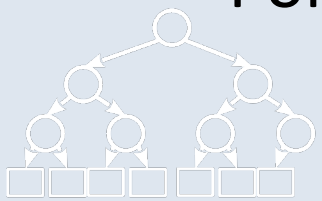
Preskočne liste – konstruiranje – histogram

- p=0.5, n=12 , h=4

$$H(k) = ceil(n \cdot (1 - p^k))$$

$$H(1) = ceil(12 \cdot (1 - 0.5^1)) = 6$$
$$H(2) = ceil(12 \cdot (1 - 0.5^2)) = 9$$
$$H(3) = ceil(12 \cdot (1 - 0.5^3)) = ceil(10.5) = 11$$
$$H(4) = ceil(12 \cdot (1 - 0.5^3)) = ceil(11.25) = 12$$

- Pokušaj izračuna za sve više razine daje $H(k) = 12$

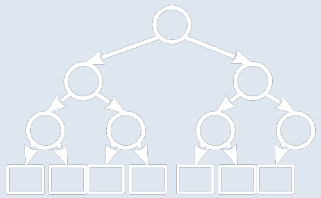


Skip lists - construction - sampling 3

- Let's create a field H of length h+1 for cum.histogram

randomLevelDirectHist()

1. |v| =1
2. r=randint(1,n) // integer from [1,n]
3. while r>H[|v|]:
    1. |v|:=|v|+1
4. return |v|

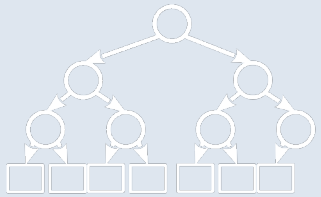


Preskočne liste – konstruiranje – uzorkovanje 3

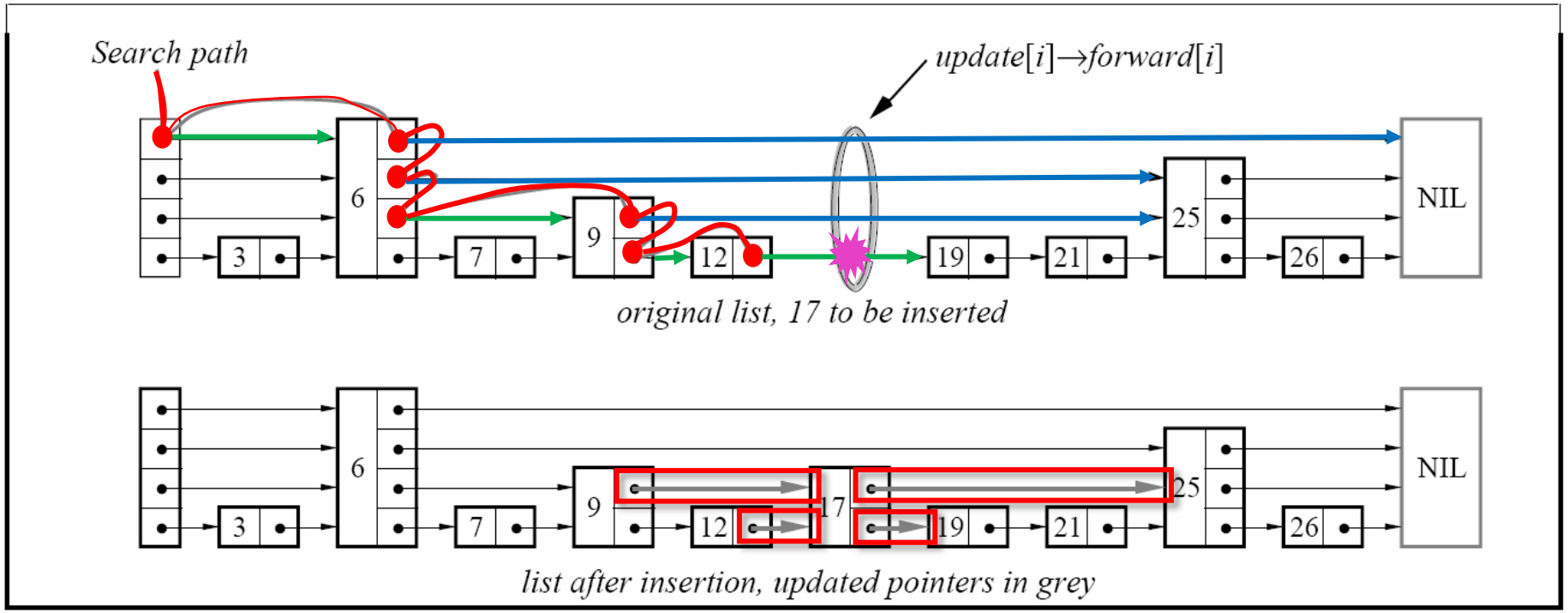
- Napravimo polje H duljine h+1 za kum.histogram

randomLevelDirectHist()

1. |v| =1
2. r=randint(1,n) // cijeli broj iz [1,n]
3. while r>H[|v|]:
    1. |v|:=|v|+1
4. return |v|

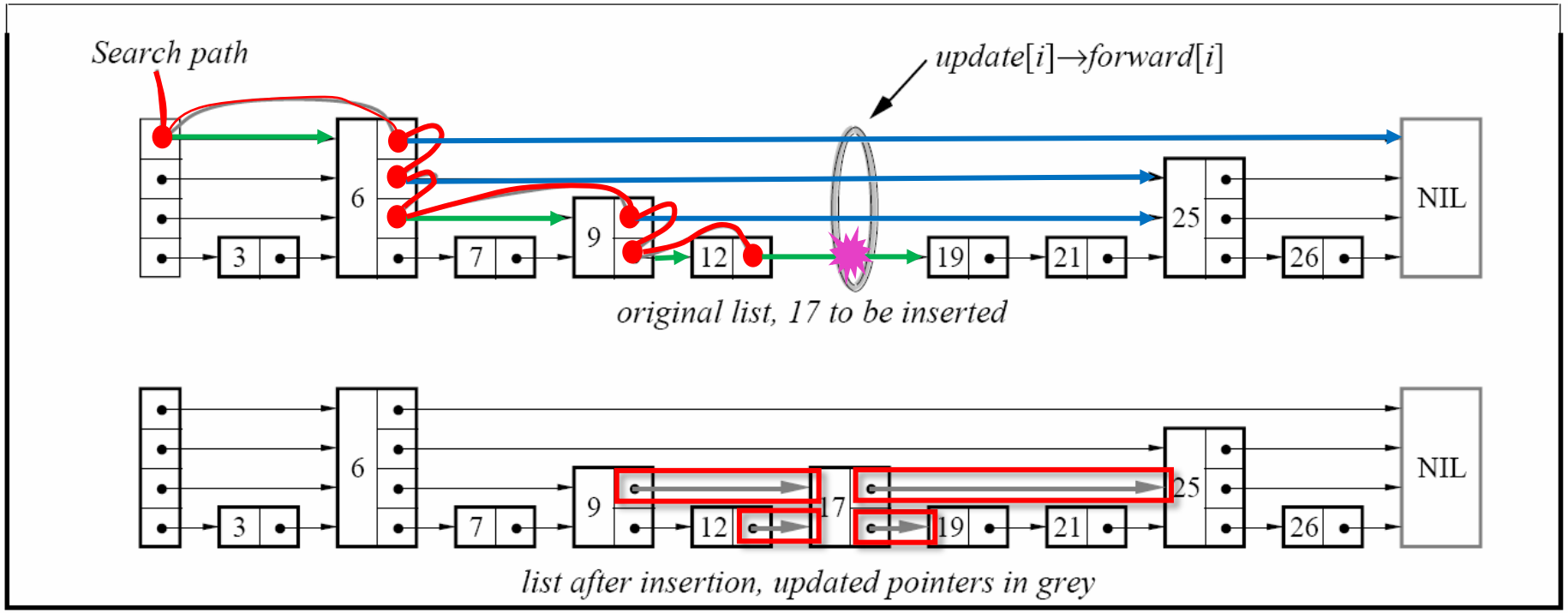


Advanced topics



Deletion?

Napredne teme

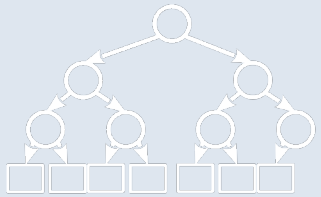


Brisanje?

Jump lists - application

- An alternative to trees for applications with high parallelism
  - Simpler implementations for lock-free operations
  - Larger memory footprint for faster access
  - Simpler insert and delete operations

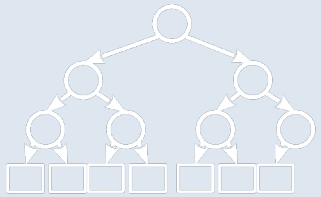
- Inspiration for algorithms
  - approximate nearest neighbors (2016)
  - the shortest routes in the road network (2015)
  - dynamic fields in quantum algorithms (2021)



Preskočne liste - primjena

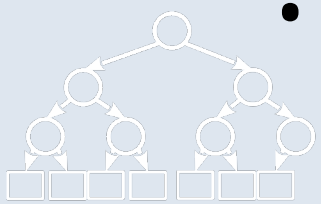
- Alternativa stablima za primjene sa velikim paralelizmom
  - Jednostavnije implementacije za lock-free operacije
  - Veće zauzeće memorije za brži pristup
  - Jednostavnije operacije umetanja i brisanja

- Inspiracija za algoritme
  - približni najbliži susjedi(2016)
  - najkraći putevi u cestovnoj mreži(2015)
  - dinamička polja u kvantnim algoritmima(2021)



Random Algorithms - Advanced Topics

- Open question BPP=P?
  - Randomness helps BUT…
    - PRNG
    - Derandomization
- Cryptography, theory of computer learning, distributed computing
  - blockchain
- Expansion and definition of terms
  - Knowledge, secrecy, learning, evidence, coincidence
- Interactive proof systems, probabilistically verifiable proofs



Nasumični algoritmi – napredne teme

- Otvoreno pitanje BPP=P?
  - Nasumičnost pomaže, ALI...
    - PRNG
    - Derandomizacija
- Kriptografija, teorija računalnog učenja, raspodijeljeno računarstvo
  - blockchain
- Proširenje i definiranje pojmova
  - Znanje, tajnost, učenje, dokaz, slučajnost
- Interaktivni sustavi dokaza, vjerojatnosno provjerljivi dokazi