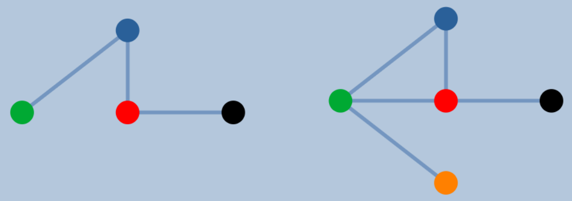
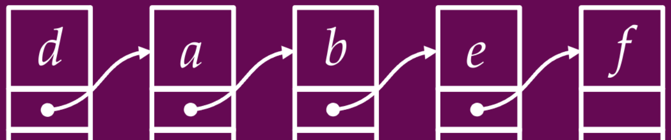
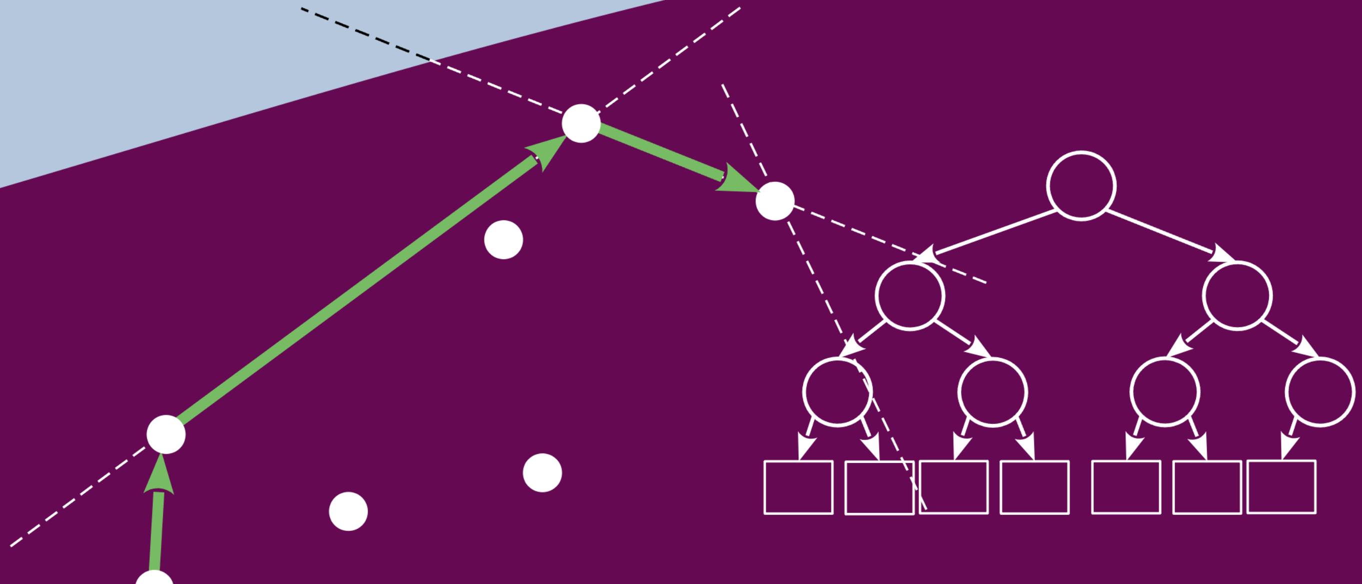
Advanced algorithms and structures data

Week 8: Random algorithms







Random algorithms

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

Lecture based on:

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

Coincidence?

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

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

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

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.

Motivation – better algorithms

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

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)

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...

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*

Random algorithms - paradigms

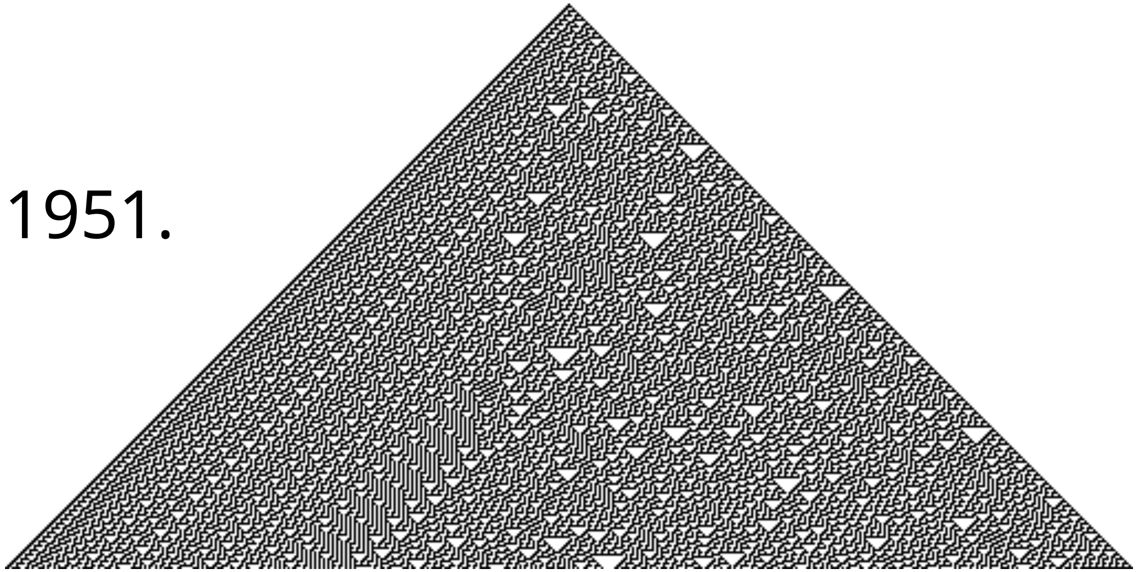
1. Deception of the opponent
2. Random sampling -> eg from the population
3. Finding "witnesses"-> eg tests for evidence
4. Fingerprinting and hashing
5. Random redeployment
6. Load balancing -> distributed computing
7. Fast-mixing Markov chains -> approximate counts
8. Isolation and breaking of symmetries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

Pseudo-random number generators

- Rich development

- Linear congruent generators (LCG) – 1951.

- [Rule 30](#) – 1983



- [Mean square and Weyl sequence generator](#) – 2017

Linear congruent generator

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

!"#= (!+)

- Constants: *a, c, m*
- "Seed”: $

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

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%

Random quicksort

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

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

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

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

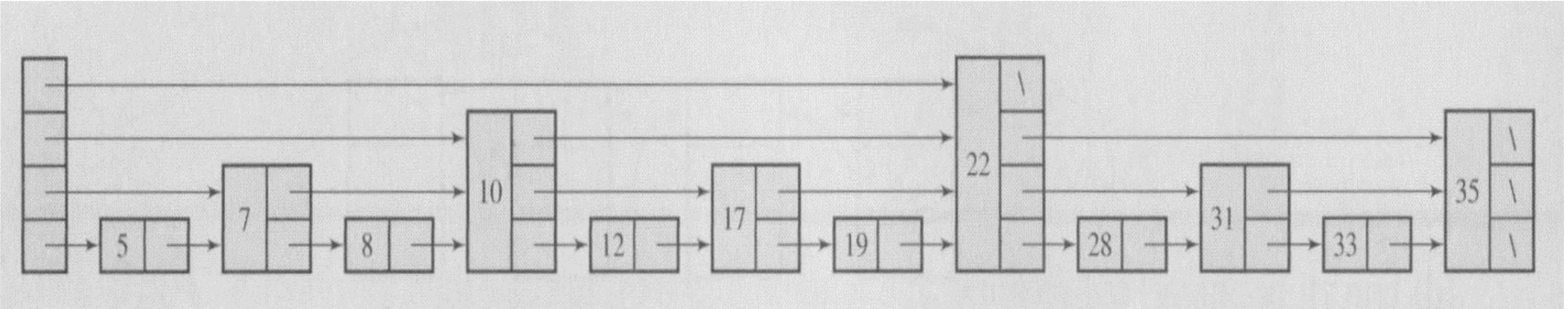
* proof in [MB95]

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!!!

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

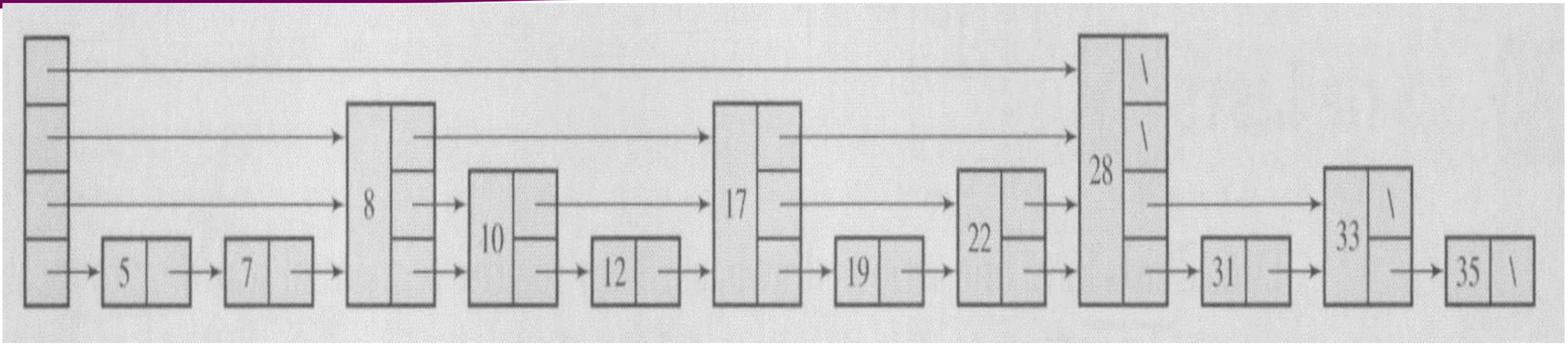


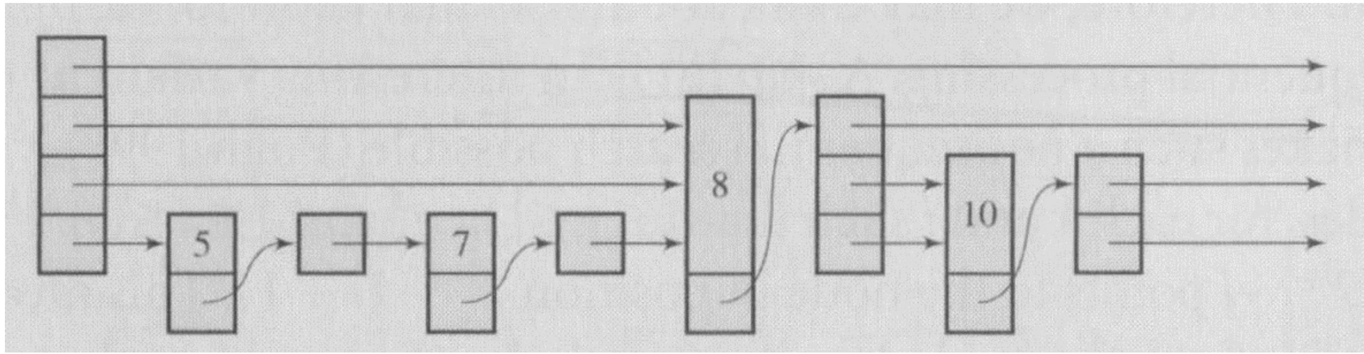
Skip lists

- **Withdrawal** from the request for proper hierarchical arrangement of nodes

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

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

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

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

Skip lists

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

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

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

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

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

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

Jump lists - construction - number of levels

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

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

$$h \leq 1 + \quad \% = 1 + \frac{1}{} \quad \$ \quad \overline{\%}$$

  - We take

$$h = \quad (1 + \quad \$ \;) \quad \overline{\%}$$

Example: p=0.5, n=12

$$h = \quad 1 + (\quad !12 = \quad) 4.6 = 4 \quad (\quad)$$

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

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
```

Skip lists - construction - sampling 2

- Direct sampling from the "cut" cumulative distribution F(k) (break at h)
    - "Surplus" is distributed at the highest level
    - $\notin(\;\;) \leq \;\; ; (h = 1) \quad (\;\;)$
      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)

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

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

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

* trivial, (0) = 0

Jump lists - construction - histogram

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

( ) = (1 (1 − 1))

(1) = 12 1 ((1 − 0.5#) = 6 (2) =
12 1 (1 − 0.5%) = 9
3 = 12 1 (1 − 0.52) = 10.5 = 11 4 = 12 1
(1 − 0.52) = 11.25 = 12

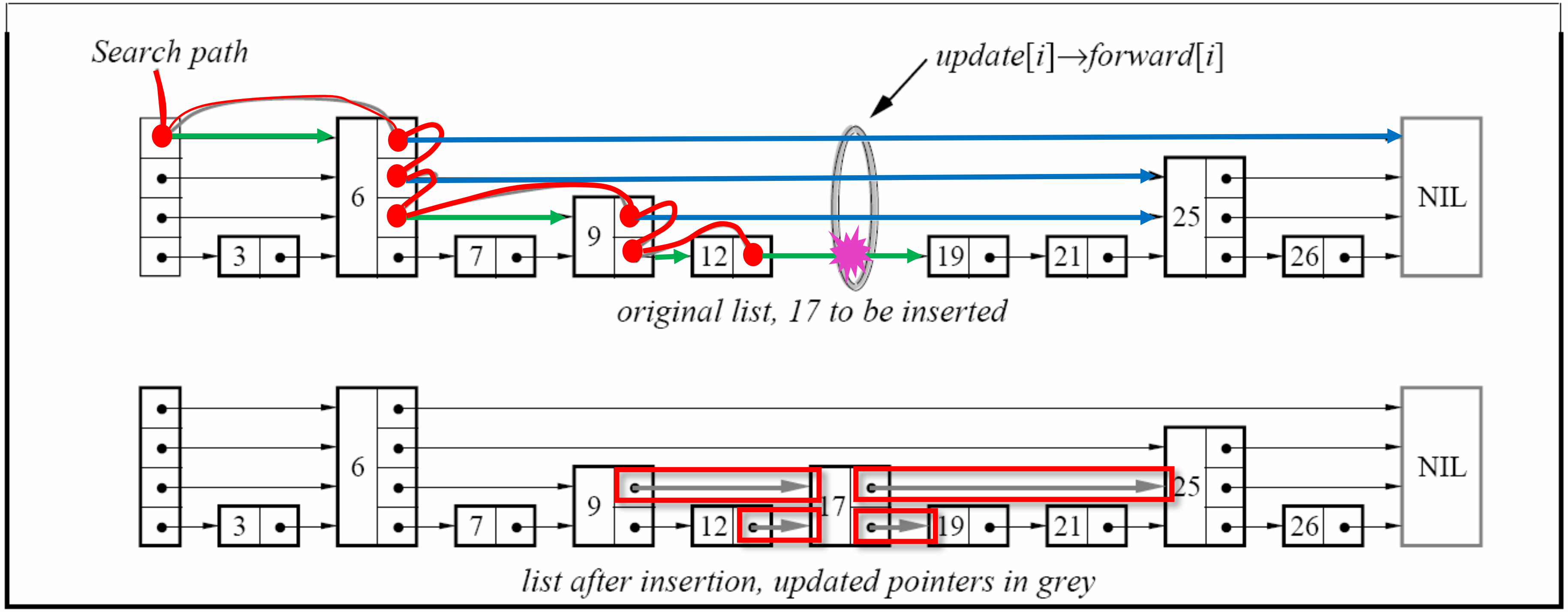
- Attempting to calculate for all higher levels gives ( ) = 12

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|
```

Advanced topics



Deletion?

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)

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