Randomized algorithms

Leonid Zhukov

School of Applied Mathematics and Information Science
National Research University Higher School of Economics

19.12.2013

Randomized algorithms

Definition

Randomized algorithm is an algorithm that makes random choices during its exectution. Its behaviour varies from one run to another even with fixed input.

Main benefits compare to deterministic algorithms: simplicity and speed

Randomized algorithms:

- Monte Carlo algorithms: guaranteed to run in fixed time (polynomial), but may produce incorrect results with bounded probability
- Las Vegas algorithms: guaranteed to produce correct results, but running time is random.

Can always convert Las Vegas $\rightarrow$ Monte Carlo by stopping at some point

Randomized algorithms

Finding an element in an array

Las Vegas algorithm:

```
finding_a_LV(array A, 'a')
    repeat
        Randomly select one element out of n elements.
    until 'a' is found
```

Monte Carlo algorithm:

```
finding_a_MC(array A, 'a', k)
    i=0
    repeat
        Randomly select one element out of n elements.
        i = i + 1
    until i=k or 'a' is found
```

Randomized algorithms

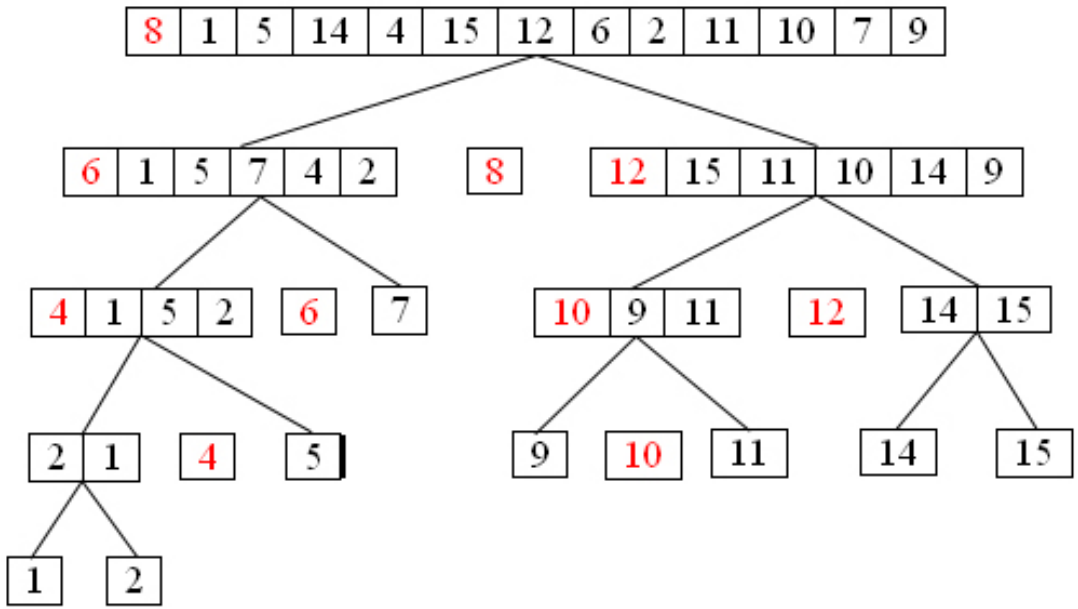
Basic principles of randomized algorithms construction (R.M. Karp)

- Foiling the adversary
- Abundance of witnesses
- Fingerprinting.
- Checking identities
- Random sampling, ordering and partitioning
- Rapidly mixing Markov chains
- Symmetry breaking

Randomized quick sort

Quick sort algorithm:

- Pick a pivot p
- Split the array into A_1 and A_2, where $A_1 < p$, $A_2 > p$
- Sort A_1 and A_2 recursevely are return A_1, p, A_2

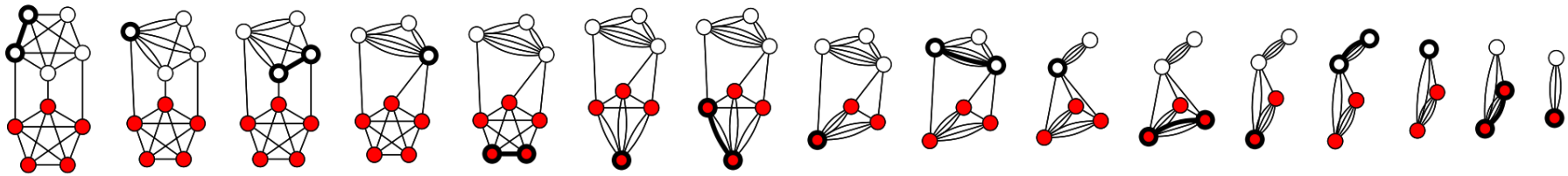


Solution: pick a pivot p uniformly at random (Las Vegas algorithm)
Run time $O(2n \log n)$

Min-Cut algorithm (Karger's Algorithm)

Global min-Cut algorithm: find a $cut(A, B)$ with miniml weight

- Pick an edge uniformly at random
- Contract selected edge, merge endpoints into one node
- Repear until two nodes left



m-edges, n-nodes

- After one run, min cut found with $P(success) \geq 2/n^2$, $P(err) \leq 1 - 2/n^2$
- After $n^2 \ln n$ independend runs min cut not found $P(err) \leq 1/n^2$
- Running time: $O(mn^2 \ln n)$, best $O(m \ln^3 n)$,

(Monte Carlo algorithm)

Verifying polynomial identities

Given two polynomials$F(x)$ and $G(x)$ degree d, verify they are identical
$F(x) = (x-7)(x-3)(x-1)(x+2)(2x+5)$
$G(x) = 2x^5 - 13x^4 - 21x^3 + 127x^2 + 121x - 210$
Multiply through: $O(d^2)$ operations, evaluating $O(d)$
Algorithm:

- Evaluate at random point $r \in \{1..R\}$ and compare results:
  if $F(r) \neq G(r)$, then $F(x) \neq G(x)$, always correct
  if $F(r) = G(r)$, then $F(x) = G(x)$ can be wrong:
  r can be a root of $Q(x) = F(x) - G(x)$ and $F(x) \neq G(x)$
- $Q(x)$- polynomial degree d, can have only d roots
- if $F(x) \neq G(x)$, probability $P[F(r) = G(r)] = d/R$
- After $k < d$ independent tests (with replacement) $P(err) \leq (d/R)^k$

(Monte Carlo algorithm)

Verifying matrix multiplication

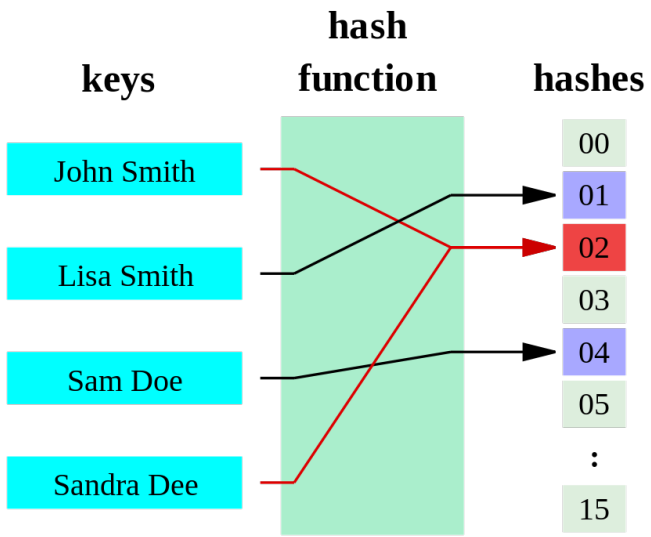
Given three matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, verify that $\mathbf{AB} = \mathbf{C}$
Matrix multiplication $O(n^3)$
Algorithm:

- Choose random vector $\mathbf{r} \in \{0, 1\}$. Compute $\mathbf{ABr} = \mathbf{A}(\mathbf{Br})$ and $\mathbf{Cr}$:
  if $\mathbf{ABr} \neq \mathbf{Cr}$, then $\mathbf{AB} \neq \mathbf{C}$, always correct
  if $\mathbf{ABr} = \mathbf{Cr}$, then $\mathbf{AB} = \mathbf{C}$ can be wrong:
  $\mathbf{r}$ can be solution of $\mathbf{Dr} = 0$, where $\mathbf{D} = \mathbf{AB} - \mathbf{C}$ and $\mathbf{AB} \neq \mathbf{C}$
- if $\mathbf{AB} \neq \mathbf{C}$, probability $P[\mathbf{ABr} = \mathbf{Cr}] \leq 1/2$
- After k trials $P(err) \leq 2^{-k}$

Running time $O(kn^2)$
(Monte Carlo algorithm)

Hashing

A hash function is any algorithm that maps data of arbitrary length to data of a fixed length - hash values (hashes).

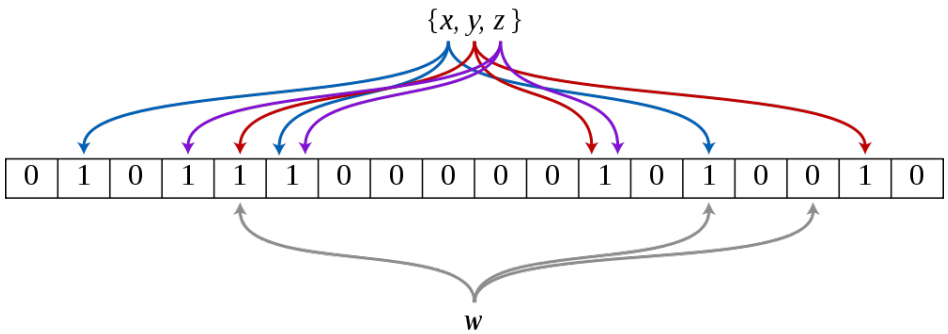


Store elements from a universe U in a table with keys (indeces) frim index space M of size $|M| = m$ and $|U| > m$, $U \to M$
Error free: big hash tables (few collisons) or chaining rules (slower processing)

Bloom filter

Bloom filter - space-efficient probabilistic data structure that is used to test weather an element is a member of a set. (does not store the items) The query returns:

- "definitely not in a set"(no false negatives)
- "possibly in the set"(possible false positive)



- empty Bloom filter is a bit array of length m filled with 0
- define k different hash functions with mapping to $1..m$
- to add an element: compute k hash values, put 1 in the positions
- to verify an element: compute k hash values:
  - if any of the bits are 0, definetly not in the set
  - if all bits are 1, probably in the set, with some probability of error

Bloom filter

- Negatives answers (element not in the set) are exact
- Positive answers (element in the set) has probability of error
  - Probability that a particular bit is still set to 0 on insertion $P = (1 - 1/m)^k$
  - After inserting n elements bit is still set to 0 with probability $P = (1 - 1/m)^{kn} \approx e^{-kn/m}$
  - Probability that a particular bit is set to 1 after n insertions, $P \approx 1 - e^{-kn/m}$
  - Probability of false positive when verifying element:

$$P(err) \approx (1 - e^{-kn/m})^k$$