Some applications of the method of moments in the analysis of algorithms

Alois Panholzer

Institute of Discrete Mathematics and Geometry
Vienna University of Technology
Alois.Panholzer@tuwien.ac.at

Universite de Paris-Nord, 16.2.2010

Outline

The Method of Moments

The Method of Moments

Motivation

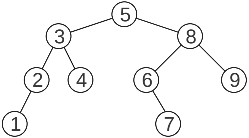
Average-case analysis of algorithms

```
procedure
  Quicksort(A:array)
  ...
end
```

E.g., **Quicksort**
input string: random permutation of size n

- number of comparisons to sort elements
- number of recursive calls to sort elements

Analysis of average behaviour of parameters in random structures



E.g., **random binary search tree** of size n

- number of leaves in tree
- depth of j-th smallest node in tree

The Method of Moments

Motivation

Average-case analysis:

X_n: parameter (i.e., random variable) under consideration for random size-n instance

- Expectation ($=$ mean value) $\mathbb{E}(X_n)$
- Concentration results, Variance $\mathbb{V}(X_n)$
- Limiting distribution results

$$X_n \xrightarrow{(d)} X, \qquad X_n \text{ converges in distribution to r.v. } X$$

- Tail estimates ("bounds on rare events")

The Method of Moments

Showing limiting distribution results

Basis: Theorem of Fréchet and Shohat

(Second central limit theorem)

If

(i) all positive r-th integer moments of X_n converge to the r-th moments of a r.v. X:

$$\mathbb{E}(X_n^r) \to \mathbb{E}(X^r), \quad \text{for all } r \geq 1$$

(ii) the distribution of X is uniquely defined by its moments

then $X_n \xrightarrow{(d)} X$, i.e., X_n converges in distribution to X

The Method of Moments

Showing limiting distribution results

This means: the distribution function $F_n(x) = \mathbb{P}\{X_n \leq x\}$ of X_n converges **pointwise** for every $x \in \mathbb{R}$ to the distribution function $F(x) = \mathbb{P}\{X \leq x\}$ of X.

Consider $X_n = \sum_{i=1}^{n} Y_{n,i}$, $Y_{n,i}$ independent identically distr. as Y, $\mathbb{P}\{Y = 1\} = \mathbb{P}\{Y = -1\} = \frac{1}{2}$.

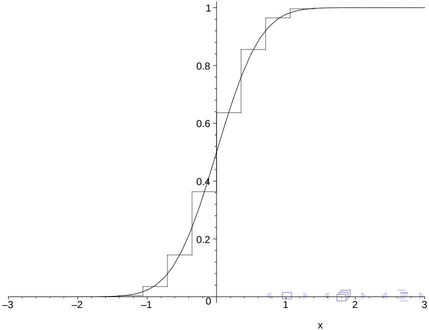
The Method of Moments

Showing limiting distribution results

This means: the distribution function $F_n(x) = \mathbb{P}\{X_n \leq x\}$ of X_n converges **pointwise** for every $x \in \mathbb{R}$ to the distribution function $F(x) = \mathbb{P}\{X \leq x\}$ of X.

Consider $X_n = \sum_{i=1}^{n} Y_{n,i}$, $Y_{n,i}$ independent identically distr. as Y, $\mathbb{P}\{Y = 1\} = \mathbb{P}\{Y = -1\} = \frac{1}{2}$.

$n = 10$:



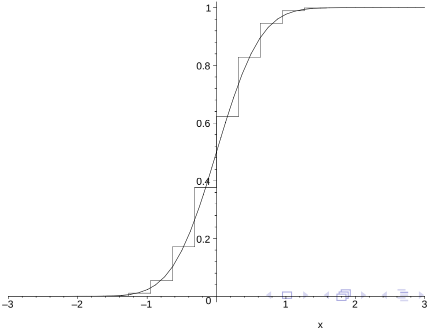
The Method of Moments

Showing limiting distribution results

This means: the distribution function $F_n(x) = \mathbb{P}\{X_n \le x\}$ of X_n converges **pointwise** for every $x \in \mathbb{R}$ to the distribution function $F(x) = \mathbb{P}\{X \le x\}$ of X.

Consider $X_n = \sum_{i=1}^{n} Y_{n,i}$, $Y_{n,i}$ independent identically distr. as Y, $\mathbb{P}\{Y = 1\} = \mathbb{P}\{Y = -1\} = \frac{1}{2}$.

$n = 20$:



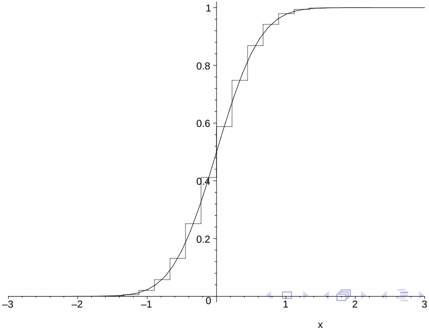
The Method of Moments

Showing limiting distribution results

This means: the distribution function $F_n(x) = \mathbb{P}\{X_n \leq x\}$ of X_n converges **pointwise** for every $x \in \mathbb{R}$ to the distribution function $F(x) = \mathbb{P}\{X \leq x\}$ of X.

Consider $X_n = \sum_{i=1}^{n} Y_{n,i}$, $Y_{n,i}$ independent identically distr. as Y, $\mathbb{P}\{Y = 1\} = \mathbb{P}\{Y = -1\} = \frac{1}{2}$.

$n = 40$:



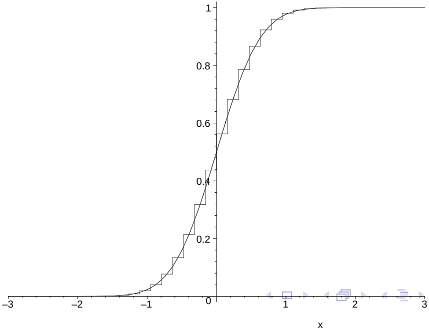
The Method of Moments

Showing limiting distribution results

This means: the distribution function $F_n(x) = \mathbb{P}\{X_n \leq x\}$ of X_n converges **pointwise** for every $x \in \mathbb{R}$ to the distribution function $F(x) = \mathbb{P}\{X \leq x\}$ of X.

Consider $X_n = \sum_{i=1}^{n} Y_{n,i}$, $Y_{n,i}$ independent identically distr. as Y, $\mathbb{P}\{Y = 1\} = \mathbb{P}\{Y = -1\} = \frac{1}{2}$.

$n = 80$:



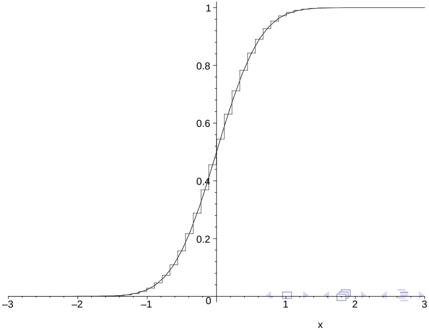
The Method of Moments

Showing limiting distribution results

This means: the distribution function $F_n(x) = \mathbb{P}\{X_n \leq x\}$ of X_n converges **pointwise** for every $x \in \mathbb{R}$ to the distribution function $F(x) = \mathbb{P}\{X \leq x\}$ of X.

Consider $X_n = \sum_{i=1}^{n} Y_{n,i}$, $Y_{n,i}$ independent identically distr. as Y, $\mathbb{P}\{Y = 1\} = \mathbb{P}\{Y = -1\} = \frac{1}{2}$.

$n = 160$:



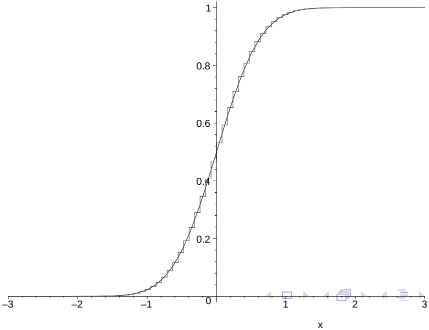
The Method of Moments

Showing limiting distribution results

This means: the distribution function $F_n(x) = \mathbb{P}\{X_n \le x\}$ of X_n converges **pointwise** for every $x \in \mathbb{R}$ to the distribution function $F(x) = \mathbb{P}\{X \le x\}$ of X.

Consider $X_n = \sum_{i=1}^{n} Y_{n,i}$, $Y_{n,i}$ independent identically distr. as Y, $\mathbb{P}\{Y = 1\} = \mathbb{P}\{Y = -1\} = \frac{1}{2}$.

$n = 320$:



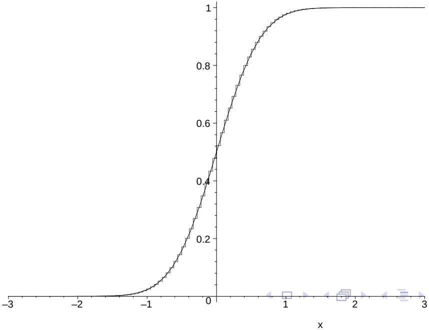
The Method of Moments

Showing limiting distribution results

This means: the distribution function $F_n(x) = \mathbb{P}\{X_n \le x\}$ of X_n converges **pointwise** for every $x \in \mathbb{R}$ to the distribution function $F(x) = \mathbb{P}\{X \le x\}$ of X.

Consider $X_n = \sum_{i=1}^{n} Y_{n,i}$, $Y_{n,i}$ independent identically distr. as Y, $\mathbb{P}\{Y = 1\} = \mathbb{P}\{Y = -1\} = \frac{1}{2}$.

$n = 640$:



The Method of Moments

Showing limiting distribution results

Point (ii) is satisfied under **growth conditions** of moments $\mathbb{E}(X^r)$

Carleman criterion:

If

$$\sum_{r\geq 1} \frac{1}{\sqrt[2r]{\mathbb{E}(X^{2r})}} = \infty,$$

then X is uniquely defined by its sequence of moments.

The Method of Moments

Applications in average-case analysis

Analysis of Algorithms and random structures:

- Often: one obtains distributional recurrences for parameters of interest
- In many cases: difficult to treat distributional recurrences directly
- But: recurrences for moments usually simpler

The Method of Moments

Applications in average-case analysis

A "typical situation":

- Recurrences for $\mathbb{E}(X_n^r)$ are linear
- They differ only in the inhomogeneous part
- Inhomogeneous part contains lower moments $\mathbb{E}(X_n^1), \ldots, \mathbb{E}(X_n^{r-1})$

If method applicable:

one can pump out successively all moments (at least asymptotically)

Example I: Total displacement in linear probing hashing

Total displacement in linear probing hashing

Problem description

Linear probing hashing

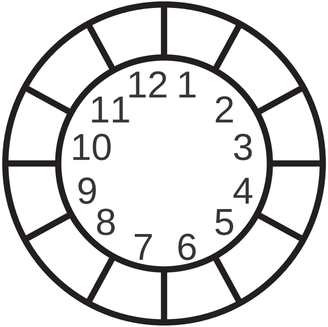
- Table of length m
- Hash function h maps keys to $[1 \dots m]$ of table addresses
- Sequences of $n \leq m$ elements entering sequentially into table
- Each element x is placed at first unoccupied location starting from $h(x)$ in cyclic order:

$$h(x), h(x)+1, \dots, m, 1, 2, \dots, h(x)-1$$

Total displacement in linear probing hashing

Problem description

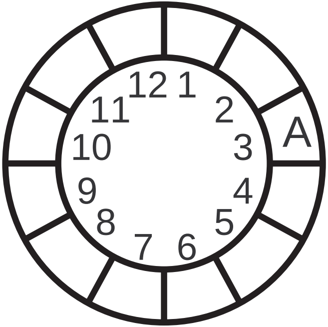
Example of constructing a hash table:



Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:

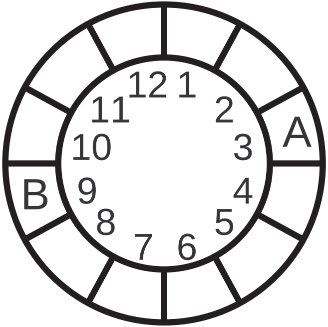


$A \ldots h(A) = 3$

Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:

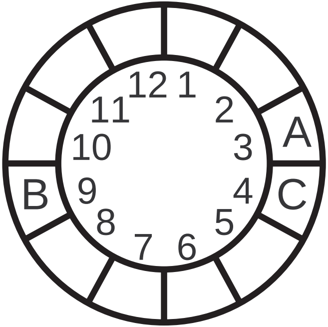


$A \dots h(A) = 3$
$B \dots h(B) = 9$

Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:

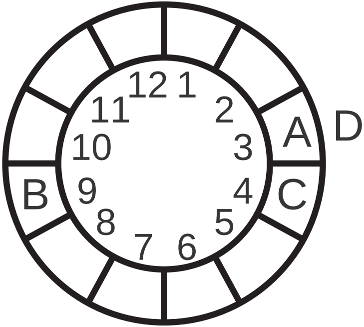


$A \ldots h(A) = 3$
$B \ldots h(B) = 9$
$C \ldots h(C) = 4$

Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:

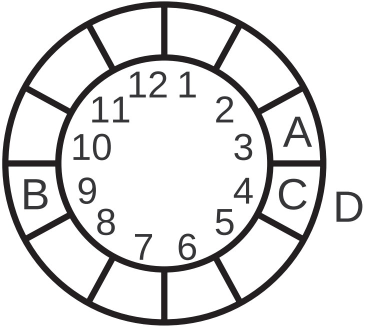


$A \dots h(A) = 3$
$B \dots h(B) = 9$
$C \dots h(C) = 4$
$D \dots h(D) = 3$

Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:

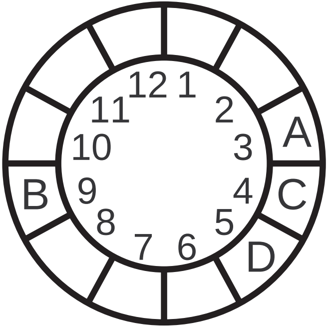


$A \ldots h(A) = 3$
$B \ldots h(B) = 9$
$C \ldots h(C) = 4$
$D \ldots h(D) = 3$

Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:

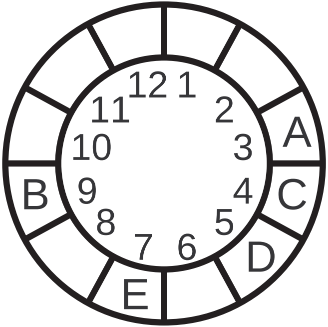


$A \dots h(A) = 3$
$B \dots h(B) = 9$
$C \dots h(C) = 4$
$D \dots h(D) = 3$

Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:

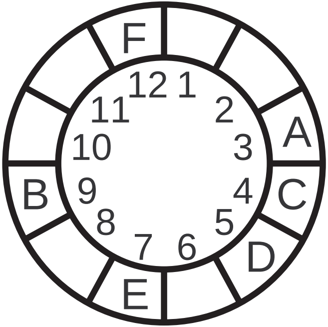


$$A \ldots h(A) = 3$$
$$B \ldots h(B) = 9$$
$$C \ldots h(C) = 4$$
$$D \ldots h(D) = 3$$
$$E \ldots h(E) = 7$$

Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:

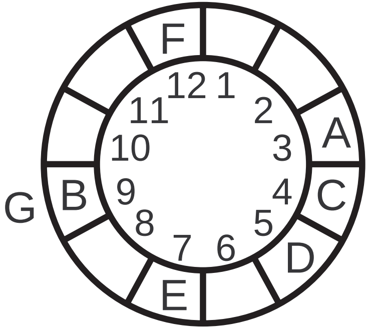


$A \dots h(A) = 3$
$B \dots h(B) = 9$
$C \dots h(C) = 4$
$D \dots h(D) = 3$
$E \dots h(E) = 7$
$F \dots h(F) = 12$

Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:

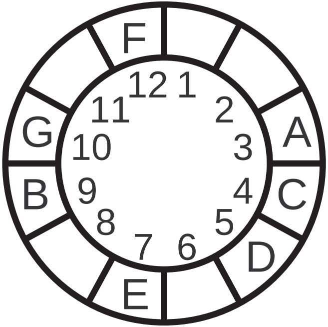


$A \dots h(A) = 3$
$B \dots h(B) = 9$
$C \dots h(C) = 4$
$D \dots h(D) = 3$
$E \dots h(E) = 7$
$F \dots h(F) = 12$
$G \dots h(G) = 9$

Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:

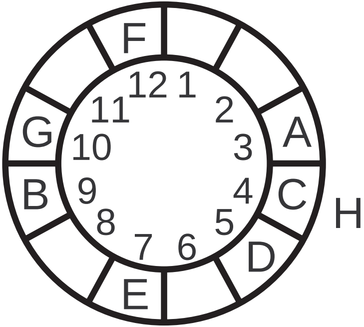


$A \ldots h(A) = 3$
$B \ldots h(B) = 9$
$C \ldots h(C) = 4$
$D \ldots h(D) = 3$
$E \ldots h(E) = 7$
$F \ldots h(F) = 12$
$G \ldots h(G) = 9$

Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:

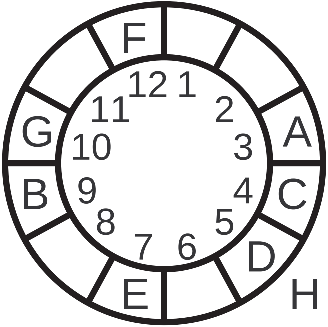


$A \ldots h(A) = 3$
$B \ldots h(B) = 9$
$C \ldots h(C) = 4$
$D \ldots h(D) = 3$
$E \ldots h(E) = 7$
$F \ldots h(F) = 12$
$G \ldots h(G) = 9$
$H \ldots h(H) = 4$

Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:

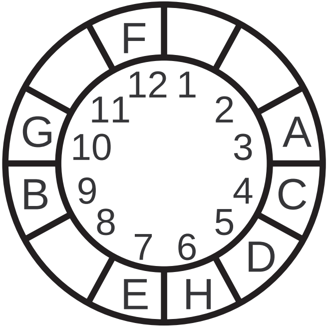


$A \ldots h(A) = 3$
$B \ldots h(B) = 9$
$C \ldots h(C) = 4$
$D \ldots h(D) = 3$
$E \ldots h(E) = 7$
$F \ldots h(F) = 12$
$G \ldots h(G) = 9$
$H \ldots h(H) = 4$

Total displacement in linear probing hashing

Problem description

Example of constructing a hash table:



$A \dots h(A) = 3$
$B \dots h(B) = 9$
$C \dots h(C) = 4$
$D \dots h(D) = 3$
$E \dots h(E) = 7$
$F \dots h(F) = 12$
$G \dots h(G) = 9$
$H \dots h(H) = 4$

Total displacement in linear probing hashing

Problem description

Displacement $d(x)$ of element x placed at location y:

circular distance between $h(x)$ and y:

$$d(x) := \begin{cases} y - h(x), & \text{if } h(x) \leq y, \\ m + h(x) - y, & \text{otherwise} \end{cases}$$

$\Rightarrow$ Costs of inserting x and searching x in table

Total displacement of sequence of n hashed values:

sum of the individual displacements

$\Rightarrow$ Construction costs of the table

Total displacement in linear probing hashing

Problem description

Assumption:

all m^n hash sequences are equally likely

$D_{m,n}$: Random variable counting the total displacement of a table of length m with n keys hashed

- Full table: $n = m$
- Almost full table: $n = m - 1$
- Sparse tables: $n = \alpha m$, load factor $0 < \alpha < 1$

Total displacement in linear probing hashing

Results

Theorem [Flajolet, Poblete and Viola, 1998]:

Result for almost full tables: the scaled random variable $\left(\frac{2}{n}\right)^{\frac{3}{2}} D_{n,n-1}$ converges in distribution to an Airy distributed random variable:

$$\left(\frac{2}{n}\right)^{\frac{3}{2}} D_{n,n-1} \xrightarrow{(d)} D,$$

where D is determined by its moments:

$$\mathbb{E}(D^r) = \frac{2\sqrt{\pi}}{\Gamma((3r-1)/2)} C_r,$$

and the constants C_r satisfy the following recurrence:

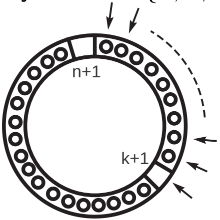
$$2C_r = (3r-4)rC_{r-1} + \sum_{j=1}^{r-1} \binom{r}{j} C_j C_{r-j}, \text{ for } r \geq 1, \quad C_0 = -1.$$

Total displacement in linear probing hashing

Proof idea

Basic decomposition of almost full tables:

- Table length $n+1$ with n elements inserted
- Before last element is inserted: Two empty cells at position $k+1$ and $n+1$
- Assumption (circular symmetry): free cell remains at $n+1$
  $\Rightarrow$ last element to be inserted has any address in $[1\ldots k+1]$
  $\Rightarrow$ displacement is any value $\in \{0, 1, \ldots, k\}$.



Total displacement in linear probing hashing

Proof idea

Decomposition leads to recursive description:

$F_{n,k}$: number of ways of creating an almost full table with n elements and total displacement k

Generating function: $F_n(q) := \sum_{k\geq 0} F_{n,k} q^k$

Recurrence:

$$F_n(q) = \sum_{k=0}^{n-1} \binom{n-1}{k} F_k(q)(1+q+\cdots+q^k)F_{n-1-k}(q)$$

Bivariate generating function: $F(z,q) := \sum_{n\geq 0} F_n(q)\frac{z^n}{n!}$

Functional equation:

$$\frac{\partial}{\partial z}F(z,q) = F(z,q)\cdot\frac{F(z,q)-qF(qz,q)}{1-q}$$

Total displacement in linear probing hashing

Proof idea

Pumping out all moments:

Generating function of r-th factorial moments:

$$f_r(z) := \frac{\partial^r}{\partial q^r} F(z,q)\Big|_{q=1}$$

$f_r(z)$ **satisfy following linear differential equation:**

$$f_r'(z)(1-T(z)) - f_r(z)\frac{T(z)(2-T(z))}{z(1-T(z))} = R_r(z),$$

where the inhomogeneous part $R_r(z)$ contains the functions $f_0(z), f_1(z), \ldots, f_{r-1}(z)$ and $T(z)$ is the tree function: $T(z) = ze^{T(z)}$

Total displacement in linear probing hashing

Proof idea

General solution:

$$f_r(z) = \frac{e^{T(z)}}{1 - T(z)} \int_0^z R_r(u) e^{-T(u)} du$$

Asymptotic behaviour around dominant singularity $z = e^{-1}$:

$$zf_r(z) \sim \frac{C_r}{(2(1-ez))^{3r/2-1/2}},$$

where constants C_r satisfy the following recurrence:

$$2C_r = (3r-4)rC_{r-1} + \sum_{j=1}^{r-1} \binom{r}{j} C_j C_{r-j}, \text{ for } r \geq 1, \quad C_0 = -1.$$

Total displacement in linear probing hashing

Proof idea

Singularity analysis of generating functions [Flajolet and Odlyzko, 1990]:

$\Rightarrow$ asymptotic equivalent of the r-th factorial and ordinary moments:

$$\left(\frac{2}{n}\right)^{\frac{3}{2}} \mathbb{E}(D^r_{n,n-1}) \to \frac{2\sqrt{\pi}}{\Gamma((3r-1)/2)} C_r$$

Total displacement in linear probing hashing

Airy distribution

Airy distribution appears in various contexts:

- Number of inversions in trees
- Path length in trees
- Area under directed lattice paths
- Counting problems for polygon models
- Number of connected graphs with n vertices and k edges
- Additive parameters in context-free grammars

"Similar" functional equations are occurring

Example II: Subtree varieties in recursive trees

Subtree varieties in recursive trees

Problem description

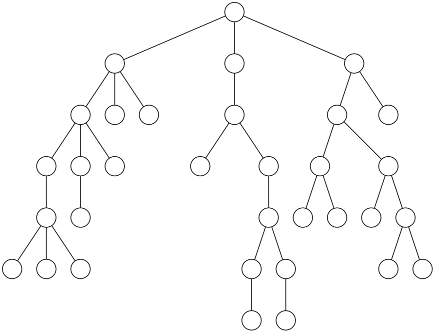
Subtree varieties in rooted trees:

- **Given:** family $\mathcal{T}$ of rooted trees
- **Consider:** random rooted tree T of size n of family $\mathcal{T}$
- **Question:** how many subtrees of T have size $k = k(n)$?

Subtree varieties in recursive trees

Problem description

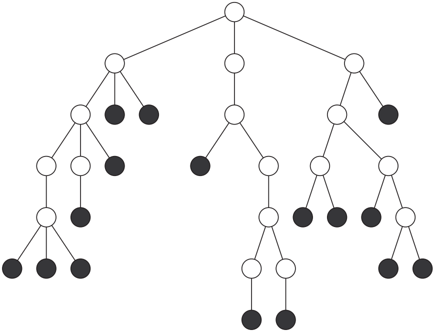
Typical situation for random tree of size n



Subtree varieties in recursive trees

Problem description

Typical situation for random tree of size n

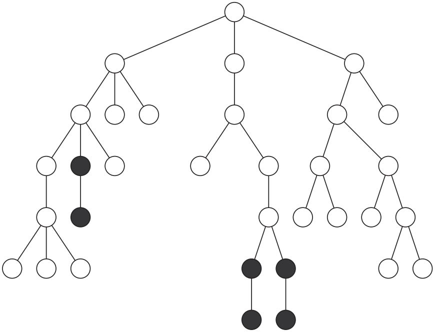


many subtrees of fixed size: size 1 (= leaves)

Subtree varieties in recursive trees

Problem description

Typical situation for random tree of size n

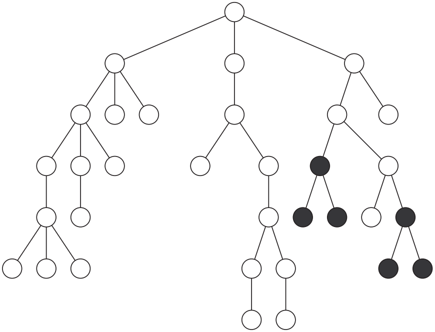


many subtrees of fixed size: size 2

Subtree varieties in recursive trees

Problem description

Typical situation for random tree of size n

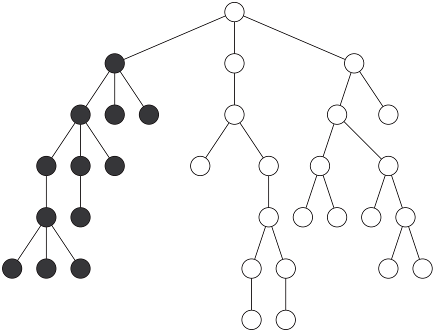


many subtrees of fixed size: size 3

Subtree varieties in recursive trees

Problem description

Typical situation for random tree of size n

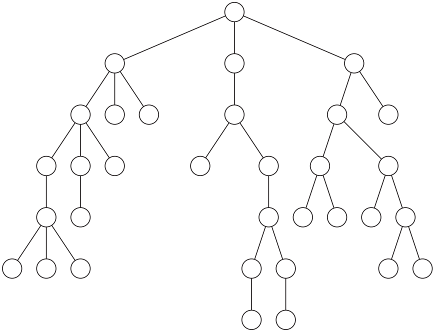


few subtrees of "large" size: size $n/3$

Subtree varieties in recursive trees

Problem description

Typical situation for random tree of size n



few subtrees of “large” size: size $n/2$

Subtree varieties in recursive trees

Recursive trees

Recursive trees:

important tree family with many applications

- models spread of epidemics
- model for pyramid schemes
- model for the family trees of preserved copies of ancient texts
- related to the Bolthausen-Sznitman coalescence model

Subtree varieties in recursive trees

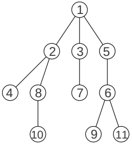
Recursive trees

Combinatorial description of a recursive tree:

- non-plane labelled rooted tree
- size-n tree labelled with labels $1, 2, \ldots, n$
- labels along path from root to arbitrary node v are increasing sequence

Random recursive trees:

all $(n-1)!$ recursive trees of size n appear with equal probability



Subtree varieties in recursive trees

Recursive trees

Simple growth rule for generating random recursive trees:

- **Step 1:** start with root labelled by 1
- **Step j:** node with label j is attached to any previous node with equal probability $1/(j-1)$

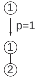
①

Subtree varieties in recursive trees

Recursive trees

Simple growth rule for generating random recursive trees:

- **Step 1:** start with root labelled by 1
- **Step j:** node with label j is attached to any previous node with equal probability $1/(j-1)$

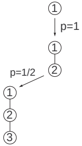


Subtree varieties in recursive trees

Recursive trees

Simple growth rule for generating random recursive trees:

- **Step 1:** start with root labelled by 1
- **Step j:** node with label j is attached to any previous node with equal probability $1/(j-1)$

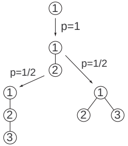


Subtree varieties in recursive trees

Recursive trees

Simple growth rule for generating random recursive trees:

- **Step 1:** start with root labelled by 1
- **Step j:** node with label j is attached to any previous node with equal probability $1/(j-1)$

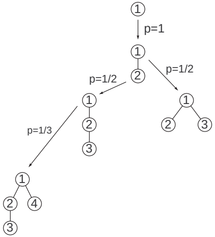


Subtree varieties in recursive trees

Recursive trees

Simple growth rule for generating random recursive trees:

- **Step 1:** start with root labelled by 1
- **Step j:** node with label j is attached to any previous node with equal probability $1/(j-1)$

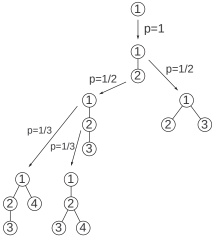


Subtree varieties in recursive trees

Recursive trees

Simple growth rule for generating random recursive trees:

- **Step 1:** start with root labelled by 1
- **Step j:** node with label j is attached to any previous node with equal probability $1/(j-1)$

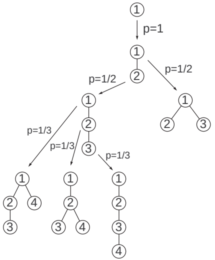


Subtree varieties in recursive trees

Recursive trees

Simple growth rule for generating random recursive trees:

- **Step 1:** start with root labelled by 1
- **Step j:** node with label j is attached to any previous node with equal probability $1/(j-1)$

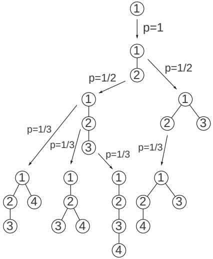


Subtree varieties in recursive trees

Recursive trees

Simple growth rule for generating random recursive trees:

- **Step 1:** start with root labelled by 1
- **Step j:** node with label j is attached to any previous node with equal probability $1/(j-1)$

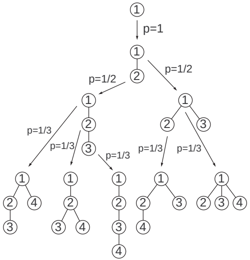


Subtree varieties in recursive trees

Recursive trees

Simple growth rule for generating random recursive trees:

- **Step 1:** start with root labelled by 1
- **Step j:** node with label j is attached to any previous node with equal probability $1/(j-1)$

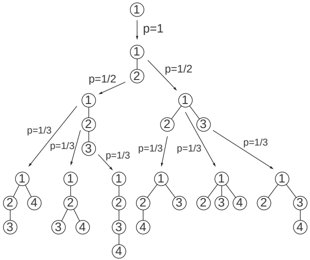


Subtree varieties in recursive trees

Recursive trees

Simple growth rule for generating random recursive trees:

- **Step 1:** start with root labelled by 1
- **Step j:** node with label j is attached to any previous node with equal probability $1/(j-1)$



Subtree varieties in recursive trees

Results

$X_{n,k}$: number of subtrees of size k in random recursive tree of size n

Theorem [Feng, Mahmoud and Pan, 2006+]:

there are three phases for behaviour of $X_{n,k}$ depending on the growth of $k = k(n)$

- subcritical case: $k/\sqrt{n} \to 0$
- critical case: $k/\sqrt{n} \to c > 0$
- supercritical case: $k/\sqrt{n} \to \infty$

Subtree varieties in recursive trees

Results

- subcritical case: $k/\sqrt{n} \to 0$:

  normalized r. v. asympt. **Gaussian distributed**

$$\frac{X_{n,k} - \frac{n}{k(k+1)}}{\sqrt{\frac{(2k^2-1)n}{k(k+1)^2(2k+1)}}} \xrightarrow{(d)} \mathcal{N}(0,1)$$

- critical case: $k/\sqrt{n} \to c > 0$:

  $X_{n,k}$ asymp. **Poisson-distributed**

$$X_{n,k} \xrightarrow{(d)} \text{Poisson}(\tfrac{1}{c^2})$$

- supercritical case: $k/\sqrt{n} \to \infty$:

  $X_{n,k}$ asymp. **denenerate**

$$X_{n,k} \xrightarrow{(d)} X, \text{ with } \mathbb{P}\{X = 0\} = 1$$

Subtree varieties in recursive trees

Proof idea

Decomposition of recursive trees according root degree:

$$\begin{aligned}\mathcal{T} &= ① \times \Big(\{\epsilon\} \,\dot{\cup}\, \mathcal{T} \,\dot{\cup}\, 1/2! \cdot \mathcal{T} * \mathcal{T} \,\dot{\cup}\, 1/3! \cdot \mathcal{T} * \mathcal{T} * \mathcal{T} \,\dot{\cup}\, \cdots\Big) \\ &= ① \times \exp(\mathcal{T})\end{aligned}$$

Generating functions: $M_k(z,v) := \sum_{n\geq 1}\sum_{m\geq 0}\mathbb{P}\{X_{n,k}=m\}\frac{z^n}{n!}v^m$

Differential equation:

$$\frac{\partial}{\partial z}M_k(z,v) = \exp\big(M_k(z,v)\big) + (v-1)z^{k-1}$$

Subtree varieties in recursive trees

Proof idea

Explicit solution of generating function:

$$M_k(z,v) = \frac{(v-1)z^k}{k} + \log\left(\frac{1}{1 - \int\limits_0^z e^{\frac{(v-1)t^k}{k}}\, dt}\right)$$

Exact solution for factorial moments:

$$\mathbb{E}(X_{n,k}^{\underline{r}}) = \frac{[\![n \ge kr+1]\!]n}{k^r} \sum_{\ell=1}^{r} \frac{\binom{n-kr-1}{\ell-1}}{\ell} \times$$

$$\times \sum_{\substack{j_1+\cdots+j_\ell=r \\ j_q\ge 1,\ 1\le q\le \ell}} \binom{r}{j_1,\ldots,j_\ell} \frac{1}{\prod_{i=1}^{\ell}(j_i k+1)}$$

Subtree varieties in recursive trees

Proof idea

Critical case: $\Rightarrow$ Asymptotically **Poisson distribed**

$$n/k^2 \to \lambda \quad \to \quad \mathbb{E}\left(X_{n,k}^r\right) \to \lambda^r$$

Subcritical case: $\Rightarrow$ Dealing with **cancellations**

Normalized r.v. $\tilde{X}_{n,k} := \frac{X_{n,k} - \mathbb{E}(X_{n,k})}{\mathbb{V}(X_{n,k})}$

$\Rightarrow$ Asymptotically **Gaussian distributed**

$$\mathbb{E}\left(\left(\frac{\tilde{X}_{n,k}}{\sqrt{\nu(k)n}}\right)^{2d}\right) \to \frac{(2d)!}{d!\,2^d}, \quad \text{for } d \geq 0,$$

$$\mathbb{E}\left(\left(\frac{\tilde{X}_{n,k}}{\sqrt{\nu(k)n}}\right)^{2d+1}\right) \to 0, \quad \text{for } d \geq 0$$

Subtree varieties in recursive trees

Remarks

Application of method of moments to asympt. Gaussian r.v.:

- heavy cancellations $\Rightarrow$ high computational effort
- method usually only "last weapon"

Subtree varieties in recursive trees

Remarks

Application of method of moments to asympt. Gaussian r.v.:

- heavy cancellations $\Rightarrow$ high computational effort
- method usually only "last weapon"

One might try first:

- analytic methods (saddle point method, continuity theorem of Levy, quasi-power theorem)
- central limit theorems for sums of independent or weakly dependent r.v.
- Stein's method
- contraction method
- martingale description

Example III: Total costs of Union-Find-algorithms

Total costs in UNION-FIND-algorithms

Problem description

UNION-FIND-problem

- Maintaining representation of equivalence classes
  (= partitions of a finite set)
- Two basic operations:
  - UNION: merge two different equivalence classes s and t into a single equivalence class
  - FIND: find equivalence class that contains a given element x

Problem arises naturally in applications in computer science
(e.g., minimum-cost spanning tree algorithms)

Total costs in Union-Find-algorithms

Problem description

Data structure for Union-Find problem, Aho et al [1974]:

- consider partition $P(S)$ of finite set S
- for every element $x \in S$: store in $R[x]$ name of the equivalence class containing x
- for every equivalence class $s \in P(S)$:
  - store in $N[s]$ the number of elements of s
  - store in $L[s]$ the elements of s in a linked list

Total costs in UNION-FIND-algorithms

Problem description

Basic algorithm for operation UNION, Yao [1976]:

“Quick Find Weighted” (QFW):

if we merge different equivalence classes s and t then we update the class with less elements:

- if $N[s] \leq N[t]$:
  set $R[x] := t$ for all x in $L[s]$
  append $L[s]$ to $L[t]$,
  set $N[t] := N[t] + N[s]$
  call new equivalence class t
- otherwise
  set $R[x] := s$ for all x in $L[t]$
  append $L[t]$ to $L[s]$
  set $N[s] := N[s] + N[t]$
  call new equivalence class s

Total costs in UNION-FIND-algorithms

Problem description

Cost of UNION-operation:

- Costs when merging equivalence classes s and t: measured by number of updated elements, i.e., the number of allocations $R[x] := s$ or $R[x] := t$
- **QFW:** cost of merging step is given by minimum of the class sizes $\min(N[s], N[t])$

Total costs in UNION-FIND-algorithms

Problem description

Basic model for sequences of UNION-operations, Yao [1976]:

Random spanning tree model:

- deal with set S of size n
- at the beginning all elements $x \in S$ are forming equivalence class $\{x\}$
- n equivalence classes will be merged into larger and larger classes by carrying out UNION-operations according following **Merging rule**

Total costs in UNION-FIND-algorithms

Problem description

Merging rule:

- choose at random a spanning tree of complete graph with vertex set S
- choose a random ordering of the edges of this spanning tree by enumerating it from 1 to $n-1$
- leads to sequence of edges $e_1 = (x_1, y_1)$, $e_2 = (x_2, y_2)$, ..., $e_{n-1} = (x_{n-1}, y_{n-1})$, with $x_i, y_i \in S$
- gives then sequence of UNION-operations

$$\textsc{Union}(R[x_1], R[y_1]),\ \textsc{Union}(R[x_2], R[y_2]),\ \ldots,\ \textsc{Union}(R[x_{n-1}], R[y_{n-1}])$$

- $\Rightarrow$ all $n^{n-2}(n-1)!$ possible sequence of UNION-operations of that kind are equally likely

Total costs in UNION-FIND-algorithms

Problem description

Total cost of algorithm QFW:

Average performance of **QFW** described by total costs:

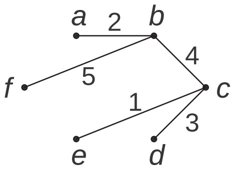
- sum of cost of every merging step when merging the elements of a set S of size n
- at beginning all elements are in different equivalence classes
- merge all elements into one equivalence class (containing all elements of S)
- carrying out sequence of $n-1$ UNION-operations according to merging rules under random spanning tree model

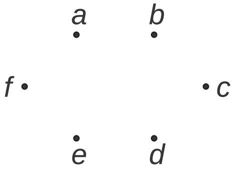
- $\Rightarrow X_n$: random variable depending only on size n of set S

Total costs in UNION-FIND-algorithms

Problem description

Example of algorithm QFW:



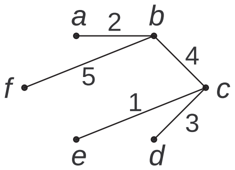


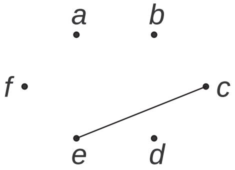
$\textsc{Union}(\{c\},\{e\}) \Rightarrow \text{Cost} = 1$

Total costs in UNION-FIND-algorithms

Problem description

Example of algorithm QFW:



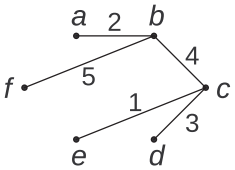


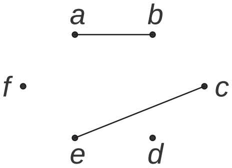
$\textsc{Union}(\{c\},\{e\}) \Rightarrow \text{Cost} = 1$
$\textsc{Union}(\{a\},\{b\}) \Rightarrow \text{Cost} = 1$

Total costs in UNION-FIND-algorithms

Problem description

Example of algorithm QFW:



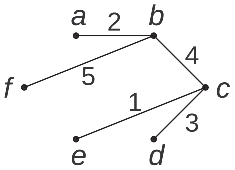


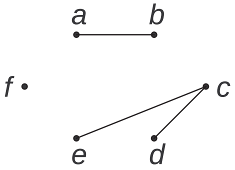
$\textsc{Union}(\{c\},\{e\}) \Rightarrow \text{Cost} = 1$
$\textsc{Union}(\{a\},\{b\}) \Rightarrow \text{Cost} = 1$
$\textsc{Union}(\{c\},\{d\}) \Rightarrow \text{Cost} = 1$

Total costs in UNION-FIND-algorithms

Problem description

Example of algorithm QFW:



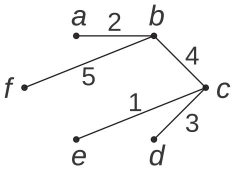


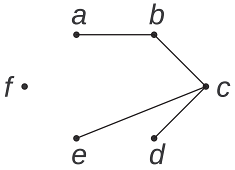
$\textsc{Union}(\{c\},\{e\}) \Rightarrow \text{Cost} = 1$
$\textsc{Union}(\{a\},\{b\}) \Rightarrow \text{Cost} = 1$
$\textsc{Union}(\{c\},\{d\}) \Rightarrow \text{Cost} = 1$
$\textsc{Union}(\{b\},\{c\}) \Rightarrow \text{Cost} = 2$

Total costs in UNION-FIND-algorithms

Problem description

Example of algorithm QFW:



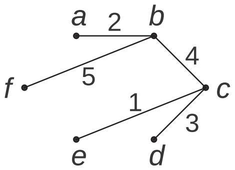


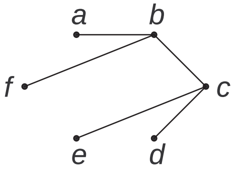
$\textsc{Union}(\{c\},\{e\}) \Rightarrow \text{Cost} = 1$
$\textsc{Union}(\{a\},\{b\}) \Rightarrow \text{Cost} = 1$
$\textsc{Union}(\{c\},\{d\}) \Rightarrow \text{Cost} = 1$
$\textsc{Union}(\{b\},\{c\}) \Rightarrow \text{Cost} = 2$
$\textsc{Union}(\{b\},\{b\}) \Rightarrow \text{Cost} = 1$

Total costs in UNION-FIND-algorithms

Problem description

Example of algorithm QFW:





UNION($\{c\}, \{e\}$) $\Rightarrow$ Cost $= 1$
UNION($\{a\}, \{b\}$) $\Rightarrow$ Cost $= 1$
UNION($\{c\}, \{d\}$) $\Rightarrow$ Cost $= 1$
UNION($\{b\}, \{c\}$) $\Rightarrow$ Cost $= 2$
UNION($\{b\}, \{b\}$) $\Rightarrow$ Cost $= 1$
Total costs $= 6$

Total costs in UNION-FIND-algorithms

Results

Theorem [Kuba and Pan, 2007]: The expectation $\mathbb{E}(X_n)$ of the total costs of the UNION-FIND-algorithm under the random spanning tree model has for $n \to \infty$ the following asymptotic expansion:

$$\mathbb{E}(X_n) = \frac{1}{\pi} n \log n + Cn + \mathcal{O}(n^{\frac{3}{4}}),$$

where the constant $C \approx 0.6315$is given as follows:

$$C = \frac{\gamma + 2\log 2}{\pi} + \sum_{n \geq 0} \frac{1}{n+1} \Big[e^{-(n+1)} \Big(R_{n+2} - R_{n+1} - \sum_{k=0}^{n} \frac{(k+1)^{k+1}}{(k+2)!} R_{n-k} \Big) - \frac{1}{\pi} \Big],$$

with

$$R_n = \sum_{k=1}^{n-1} \frac{k^k (n-k)^{n-k-1}}{k!(n-k)!} \min(k, n-k).$$

Total costs in UNION-FIND-algorithms

Results

Theorem [Kuba and Pan, 2007]: The suitably shifted and scaled r.v. X_n converges in distribution to a r.v. X, which can be characterized by its r-th integer moments:

$$\frac{X_n - \frac{1}{\pi} n \log n - Cn}{n} \xrightarrow{(d)} X, \quad \text{with} \quad \mathbb{E}(X^r) = m_r,$$

where m_r is given recursively as follows:

$$m_r = \frac{\Gamma(r-1)}{2\sqrt{\pi}\,\Gamma(r-\frac{1}{2})} \sum_{\substack{r_1+r_2+r_3=r,\\ r_2,r_3<r}} \binom{r}{r_1, r_2, r_3} m_{r_2} m_{r_3} I_{r_1,r_2,r_3}, \quad \text{for } r \geq 2,$$

with initial values $m_0 = 1$ and $m_1 = 0$ and

$$I_{r_1,r_2,r_3} = \int_0^1 \Big(\frac{1}{\pi}\big(x \log x + (1-x)\log(1-x)\big) + \min(x, 1-x)\Big)^{r_1} x^{r_2 - \frac{1}{2}} (1-x)^{r_3 - \frac{3}{2}}\, dx.$$

Total costs in UNION-FIND-algorithms

Proof idea

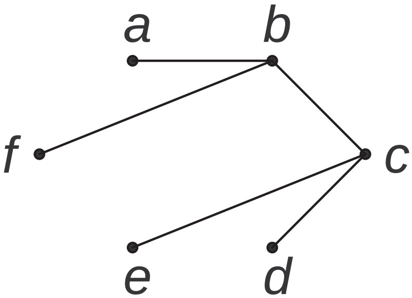
The reverse process: destroying a tree

- Start with a random spanning tree of size n
- Remove successively edges at random from remaining edges
- In every step split a connected component into two parts
- Cost of a cut is the size of the smaller part after the splitting step
- Stop when all nodes are isolated

Total costs in UNION-FIND-algorithms

Proof idea

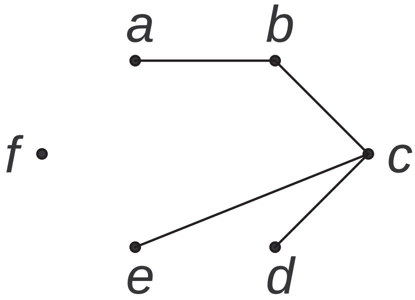
Example of destroying a tree:



Total costs in UNION-FIND-algorithms

Proof idea

Example of destroying a tree:

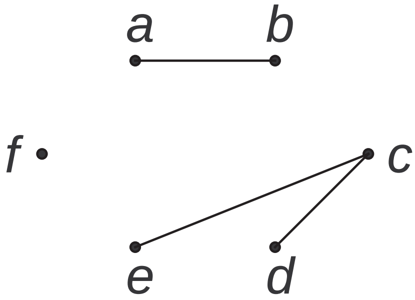


Cost = 1

Total costs in UNION-FIND-algorithms

Proof idea

Example of destroying a tree:



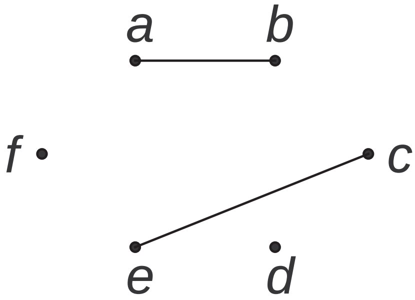
Cost = 1

Cost = 2

Total costs in UNION-FIND-algorithms

Proof idea

Example of destroying a tree:

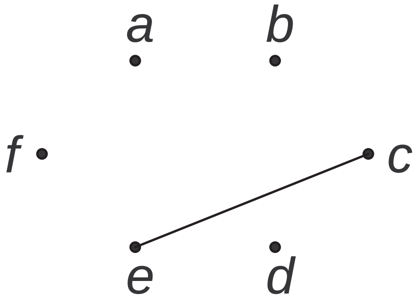


Cost = 1
Cost = 2
Cost = 1

Total costs in UNION-FIND-algorithms

Proof idea

Example of destroying a tree:

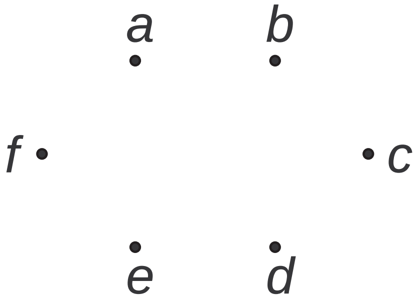


Cost = 1
Cost = 2
Cost = 1
Cost = 1

Total costs in Union-Find-algorithms

Proof idea

Example of destroying a tree:



Cost $= 1$
Cost $= 2$
Cost $= 1$
Cost $= 1$
Cost $= 1$
Total costs $= 6$

Total costs in UNION-FIND-algorithms

Proof idea

Recursive description of total costs X_n:

Distributional recurrence for rooted trees:

$$X_n \stackrel{(d)}{=} X_{S_n} + X^*_{n-S_n} + t_{n,S_n}$$

S_n: size of subtree containing root after removing random edge of randomly chosen labeled rooted tree of size n

Toll function: $t_{n,k} = \min(k, n-k)$

S_n is distributed as follows:

$$\mathbb{P}\{S_n = k\} = \frac{k T_k T_{n-k}}{(n-1) T_n},$$

with $T_n := \frac{n^{n-1}}{n!}$

Total costs in Union-Find-algorithms

Proof idea

Recurrence for r-th moments of X_n

Linear recurrence for $\mu_n^{[r]} := \mathbb{E}(X_n^r)$:

$$(n-1)\,T_n \mu_n^{[r]} = \sum_{k=1}^{n-1} k T_k \, T_{n-k} (\mu_k^{[r]} + \mu_{n-k}^{[r]}) + R_n^{[r]},$$

where the inhomogeneous part $R_n^{[r]}$ depends on the lower order moments $\mu_n^{[1]}, \ldots, \mu_n^{[r-1]}$

Total costs in UNION-FIND-algorithms

Proof idea

Generating functions treatment

Linear differential equation:

$$z(1 - T(z))C_r'(z) - (1 + zT'(z))C_r(z) = R_r(z),$$

where the inhomogeneous part depends on the g.f. $C_1(z), \ldots, C_r(z)$ for lower moments

Solution:

$$C_r(z) = \frac{T(z)}{1 - T(z)} \int_0^z \frac{R_r(t)}{tT(t)} dt$$

Asymptotic equivalents of r-th moments:

"pumped out" inductively

Total costs in UNION-FIND-algorithms

Remark

Problems of similar "nature":

- Quicksort: number of comparisons
- Pathlengths in search tree models
- Wiener-index of certain tree models

Limiting distribution characterized by "complicated" moment's sequence

Counterexample

Counterexample

Cutting down recursive trees

Cutting down procedure for rooted trees:

INPUT: tree T

$\mathtt{steps} \leftarrow 0$
$\mathtt{while}\ |T| > 1$ do
  cut off an edge e of T
  $T \leftarrow$ subtree containing the root
  $\mathtt{steps} \leftarrow \mathtt{steps} + 1$

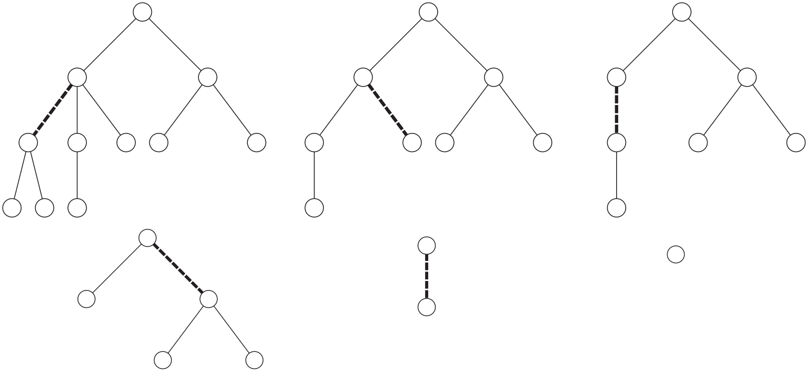
OUTPUT: steps

Remove edges until root is isolated

Counterexample

Cutting down recursive trees

An example of cutting a tree:



Size-11 tree destroyed in 5 steps.

Counterexample

Cutting down recursive trees

How many steps are done, until root is isolated?

Probability model:

- **Randomized cutting down** procedure:
  Edges in tree chosen at random in each step.
- **Random tree model** for certain tree families.

R. v. X_n **counts steps** done to destroy size-n tree.

Counterexample

Cutting down recursive trees

Why are the number of cuts to destroy the tree of interest?

- Strong connections to **coalescent models** $\Rightarrow$ theoretical physics, mathematical biology
- Cayley-trees: additive Marcus-Lushnikov process
- Recursive trees: Bolthausen-Sznitman coalescent
- X_n for recursive trees: number of collision events in the coalescent model until there is just a single block

Counterexample

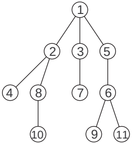
Cutting down recursive trees

Apply cutting-down procedure to recursive trees:

- non-plane labelled rooted tree
- size-n tree labelled with labels $1, 2, \ldots, n$
- labels along path from root to arbitrary node v are increasing sequence

Random recursive trees:

all $(n-1)!$ recursive trees of size n appear with equal probability



Counterexample

Cutting down recursive trees

Idea: apply recursive approach:

$$\mathbb{P}\{X_n = m\} = \sum_{k=1}^{n-1} p_{n,k}\, \mathbb{P}\{X_k = m-1\}.$$

$p_{n,k}$: Probability, that subtree containing root has size k, if we cut off random edge in random size-n tree.

Counterexample

Cutting down recursive trees

Idea: apply recursive approach:

$$\mathbb{P}\{X_n = m\} = \sum_{k=1}^{n-1} p_{n,k}\, \mathbb{P}\{X_k = m-1\}.$$

$p_{n,k}$: Probability, that subtree containing root has size k, if we cut off random edge in random size-n tree.

Attention:

- approach only applicable if randomness is preserved by cutting off random edge
- satisfied, e.g, by recursive trees, Cayley-trees, planted plane trees, d-ary trees
- not satisfied, e.g., by Motzkin-trees, binary search trees

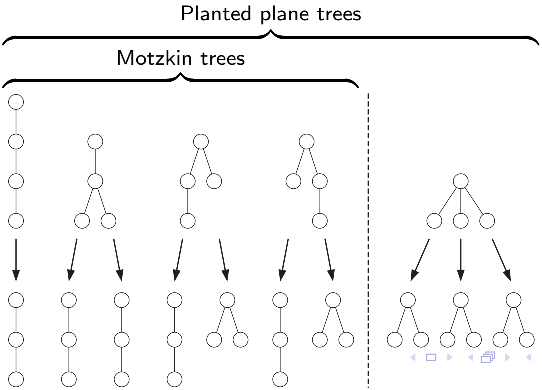
Counterexample

Cutting down recursive trees

Cutting off random edge:

Planted plane trees: randomness **preserved**

Motzkin trees: randomness **not** preserved



Counterexample

Cutting down recursive trees

Computations for recursive trees:

Splitting probability: size-n tree $\longrightarrow$ size-k tree:

$$p_{n,k} = \frac{n}{(n-1)(n-k)(n-k+1)}.$$

Recurrence:

$$\mathbb{P}\{X_n = m\} = \sum_{k=1}^{n-1} \frac{n}{(n-1)(n-k)(n-k+1)} \mathbb{P}\{X_k = m-1\}.$$

Counterexample

Cutting down recursive trees

Computations for recursive trees:

Proper **generating function**:

$$M(z,v) = \sum_{n \geq 1} \sum_{0 \leq m \leq n} \mathbb{P}\{X_n = m\} \frac{z^n}{n} v^m.$$

Differential equation:

$$\frac{\partial}{\partial z} M(z,v) = \frac{1}{z - v\left(z - (1-z)\log\left(\frac{1}{1-z}\right)\right)} M(z,v).$$

Counterexample

Cutting down recursive trees

Computations for recursive trees:

Solution of DE:

$$M(z,v) = z\, e^{\int\limits_{t=0}^{z} \frac{v\left(t-(1-t)\log\left(\frac{1}{1-t}\right)\right)}{t\left(t-v\left(t-(1-t)\log\left(\frac{1}{1-t}\right)\right)\right)}\, dt}.$$

Try method of moments:

Counterexample

Cutting down recursive trees

Computations for recursive trees:

Solution of DE:

$$M(z,v) = z\, e^{\int\limits_{t=0}^{z} \frac{v\left(t-(1-t)\log\left(\frac{1}{1-t}\right)\right)}{t\left(t-v\left(t-(1-t)\log\left(\frac{1}{1-t}\right)\right)\right)}\, dt}.$$

Try method of moments:

r-th **moments**:

$$\mathbb{E}(X_n^r) = \frac{n^r}{\log^r n} + \frac{n^r}{\log^{r+1} n}\big((r+1)H_r - r\gamma\big) + \mathcal{O}\Big(\frac{n^r}{\log^{r+2} n}\Big).$$

Scaling does not lead to a limiting distribution!

Counterexample

Cutting down recursive trees

Computations for recursive trees:

r-th **centered** moments:

$$\mathbb{E}\Big(\big(X_n - \mathbb{E}(X_n)\big)^r\Big) \sim \frac{(-1)^r}{(r-1)r} \frac{n^r}{\log^{r+1} n}, \quad r \geq 2.$$

Also centering and scaling does not lead to a limiting distribution!

Counterexample

Cutting down recursive trees

Computations for recursive trees:

r-th **centered** moments:

$$\mathbb{E}\Big(\big(X_n - \mathbb{E}(X_n)\big)^r\Big) \sim \frac{(-1)^r}{(r-1)r}\,\frac{n^r}{\log^{r+1} n}, \quad r \geq 2.$$

Also centering and scaling does not lead to a limiting distribution!

Method of moments not applicable!

Counterexample

Cutting down recursive trees

Theorem (Drmota, Iksanov, Möhle and Rösler, 2009)

The random variable

$$Y_n = \frac{X_n - \frac{n}{\log n} - \frac{n \log\log n}{(\log n)^2}}{\frac{n}{(\log n)^2}}$$

converges in distribution to a **stable random variable** Y *with characteristic function*

$$\phi_Y(\lambda) = \mathbb{E}\left(e^{i\lambda Y}\right) = e^{i\lambda \log|\lambda| - \frac{\pi}{2}|\lambda|}.$$

The moments of the limiting distribution Y do not exist!