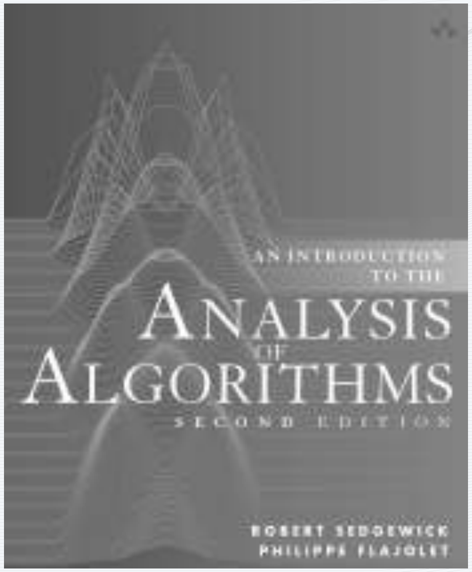


http://aofa.cs.princeton.edu

9. Words and Mappings

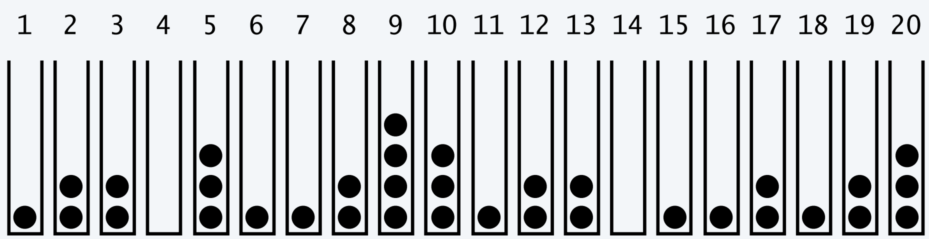
- Words
- Birthday problem
- Coupon collector problem
- **Hash tables**
- Mappings

Balls and urns

N rolls of an *M*-sided die, count number of occurrences of each value.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
9	12	19	3	5	20	10	17	16	20	13	8	2	13	9	2	15	17	3	9	11	7	18	2	10	1	20	12	10	8	19	5	5	9

N balls
M urns



Classical *occupancy* problems for an *M*-word of length *N*:

- Q. Probability that no urn has more than one ball ?
- Q. Probability that no urn is empty ?
- Q. How many empty urns ?
- Q. How many urns with *k* balls ?

Occupancy distribution (classical)

N balls, M urns
Pr {given urn has k balls}

Theorem. The probability that a value occurs k times in a random M-word of length N is

$$\binom{N}{k}\left(\frac{1}{M}\right)^k\left(1-\frac{1}{M}\right)^{N-k}$$

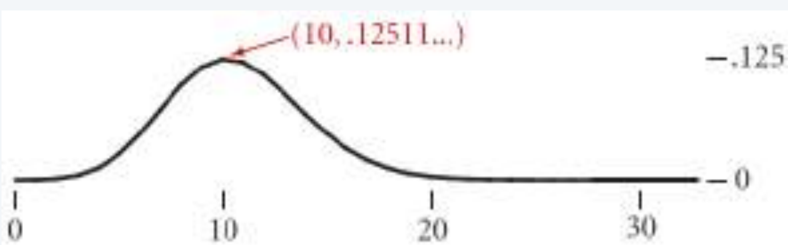
Binomial distribution

For $\alpha = N/M$ fixed and $k = O(1)$, this is

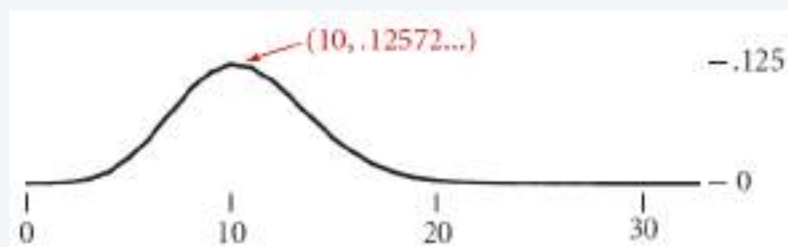
$$\frac{\alpha^k e^{-\alpha}}{k!} + o(1)$$

Poisson approximation

Proof. [See Lecture 4.]



Binomial distribution for $N = 10^4$ and $M = 10^3$ ($\alpha = 10$).



Poisson approximation for $N = 10^4$ and $M = 10^3$ ($\alpha = 10$).

Application: Hashing algorithms

Goal: Provide efficient ways to

- *Insert* key-value pairs in a *symbol table*.
- *Search* the table for the pair corresponding to a given key.

Strategy

- Develop a *hash function* that maps each key into value between 0 and $M-1$.
- Develop a *collision strategy* to handle keys that hash to the same value.

Basic algorithms (stay tuned)

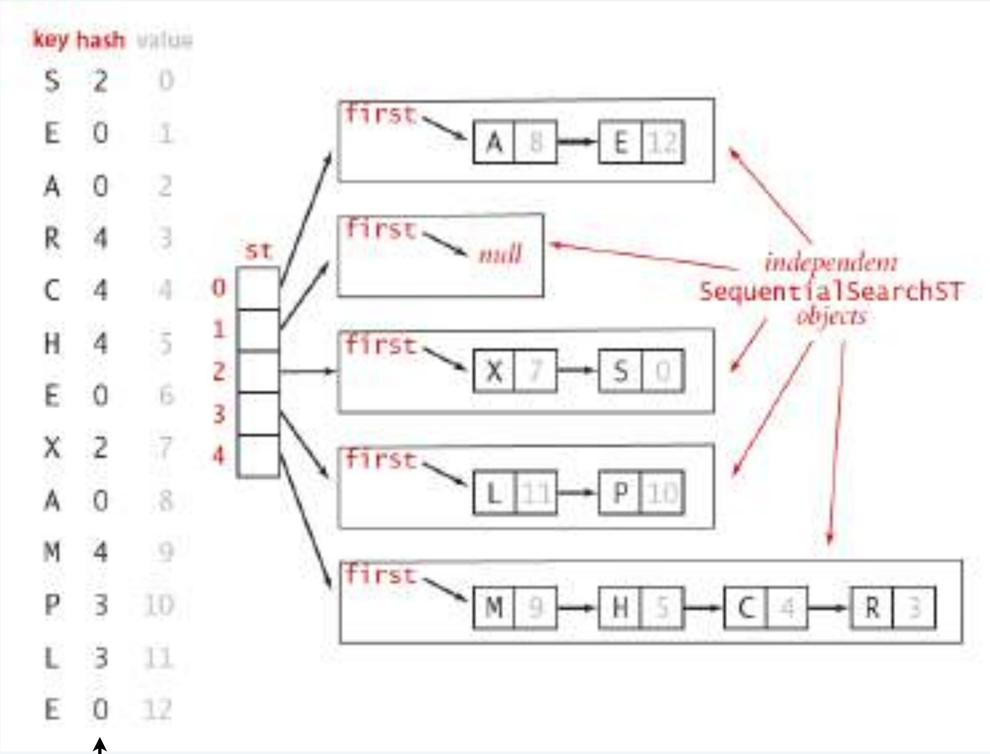
- *Separate chaining*—keep M linked lists, one for each hash value.
- *Linear probing*—use an array and scan for empty spots on collision.

Model

- *Uniform hashing assumption*—hash function maps each key in to a *random* value.

Application: Hashing with separate chaining

Keep M linked lists, one for each hash value.



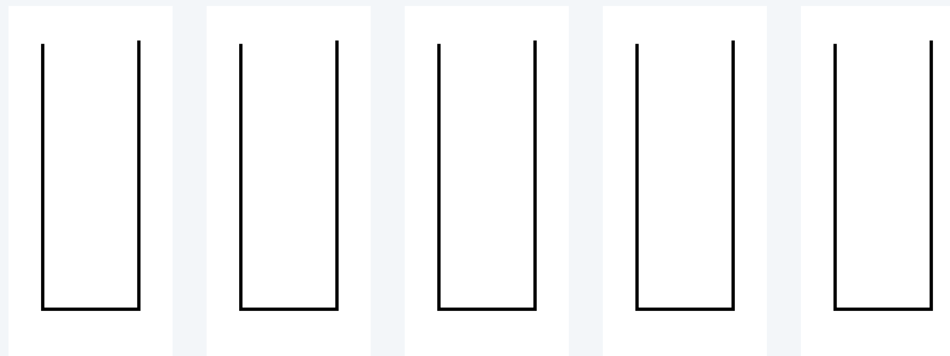
a random word!

Section 3.4

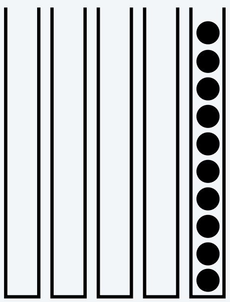
Application: Hashing with separate chaining

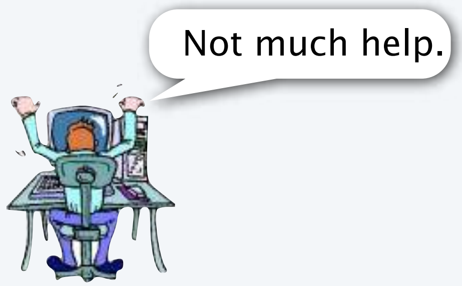
Throw N balls into M urns, one at a time.

Q. Average number of balls in each urn ?



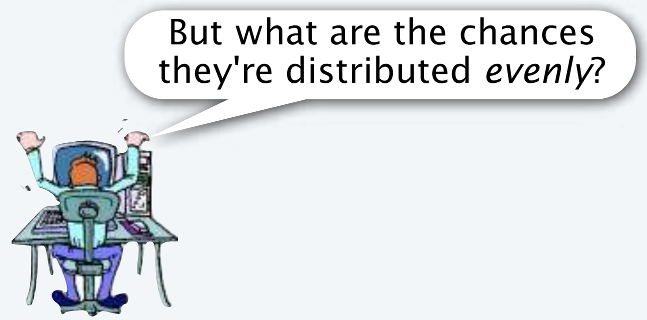
A. Obvious: N/M





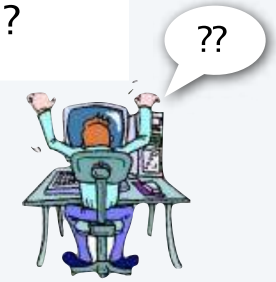
Q. Probability that a given urn has k balls ?

A. $\sim \dfrac{\alpha^k e^{-\alpha}}{k!}$ where $\alpha = N/M$



Application: Hashing with separate chaining

Q. If I make sure that $N/M < \alpha$, then the average number of probes for a search is $< \alpha$.
What is the chance that a search will use more than 5α probes (under the UHA) ?



Stirling's formula

A. $$\sum_{k>5\alpha} \frac{\alpha^k e^{-\alpha}}{k!} = e^{-\alpha} \sum_{k>5\alpha} \frac{\alpha^k}{k!} < e^{-\alpha} \sum_{k>5\alpha} \frac{\alpha^k}{(k/e)^k} < e^{-\alpha} \sum_{k>5\alpha} \frac{(e\alpha)^k}{(5\alpha)^k}$$

$$< 2e^{-\alpha} \left(\frac{e}{5}\right)^{5\alpha}$$

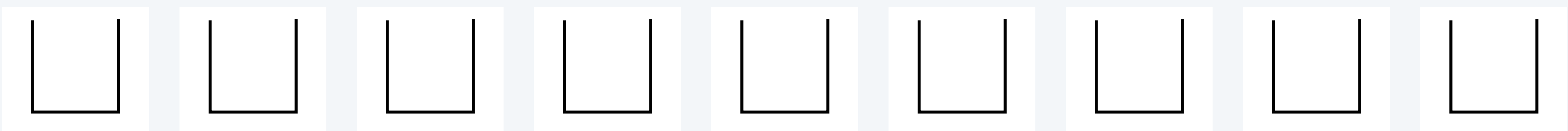
```
% bc - l
scale = 20
2*e(-10)*e(50)/5^50
.00000000000000000530
```

A. For α=10, less than .00000000000000000054

Application: Hashing with linear probing

Throw N balls into M urns, one at a time.

Resolve collisions by moving right one urn.



Q. Average number of collisions ?

Application: Hashing with linear probing

Goal: Provide efficient ways to

- *Insert* key-value pairs in a *symbol table*.
- *Search* the table for a given key.

Strategy

- Use a *hash function* as with separate chaining.
- Maintain a table size M that holds $N<M$ pairs.
- Probe the next position in the table on collision.

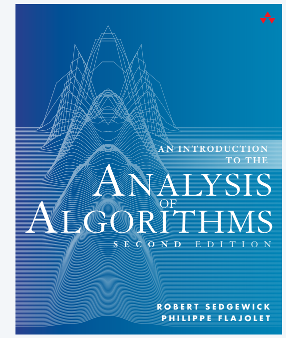
Q. Average number of probes to find one of N keys?

A. $$\sum_{k\geq 0} \frac{N}{M}\frac{N-1}{M}\cdots\frac{N-k+1}{M}$$ (Knuth, 1962) ← Difficult proof
Landmark result

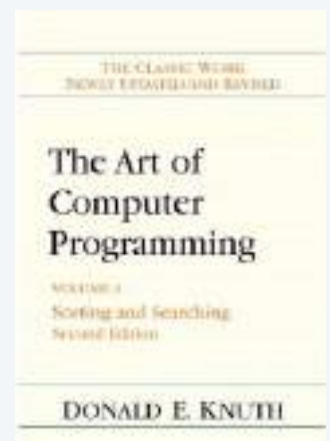
$$= Q(M) \sim \sqrt{\pi M/2}$$ when table is full ($N = M-1$)

$$\sim \frac{1}{1-\alpha}$$ when table is reasonably sparse (N/M is not close to 1)

Section 3.4



pp. 509—518



A footnote

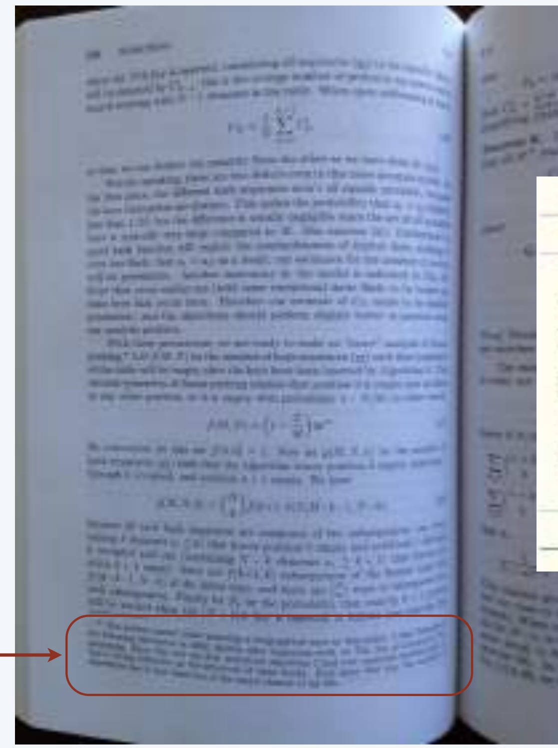
Q. Average number of probes to find one of N keys?

A. $$\sum_{k \geq 0} \frac{N}{M} \frac{N-1}{M} \cdots \frac{N-k+1}{M}$$ (Knuth, 1962)

The only footnote in Knuth's books (p. 529 vol. 3):

"The author cannot resist inserting a biographical note at this point: I first formulated the following derivation in 1962 ... Since this was the first nontrivial algorithm I had ever analyzed satisfactorily, it had a strong influence on the structure of these books."

The origin of the analysis of algorithms



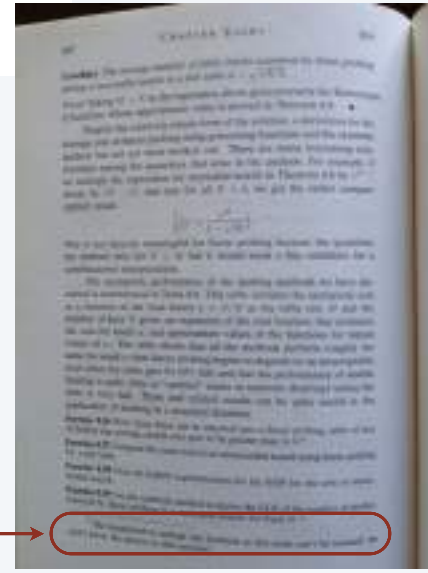
Another footnote

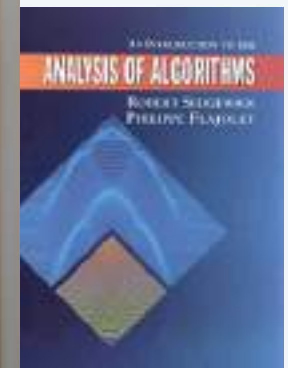
Exercise 8.39 Use the symbolic method to analyze linear probing*.

The only footnote in Sedgewick-Flajolet (p. 452):

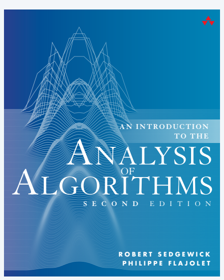
"The temptation to include one footnote at this point can't be resisted: *We don't know the answer to this exercise!*"

A challenge to students and researchers





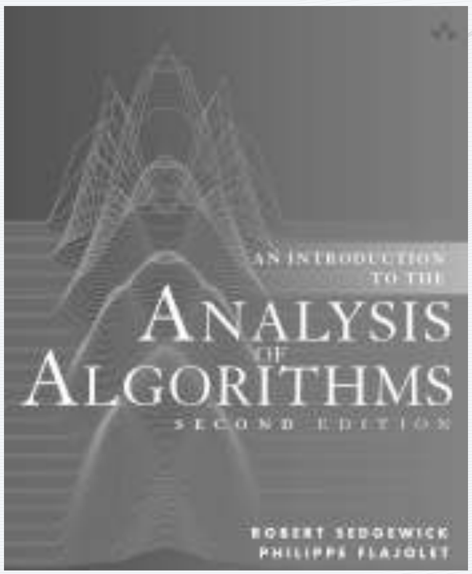
p. 452



p. 518

A. Deep connections to properties of random graphs, tree inversions, gambler's ruin, path length in trees, properties of mappings, and other classic problems. Explained by an Airy law.

Linear probing and graphs (Knuth, 1997)
On the analysis of linear probing hashing (Flajolet, Viola, and Poblete, 1997)



http://aofa.cs.princeton.edu

9. Words and Mappings

- Words
- Birthday problem
- Coupon collector problem
- Hash tables
- **Mappings**

Mappings

Q. How many *N*-words of length *N* ?

1

$M_1 = 1$

1	1
1	2
2	1
2	2

$M_2 = 4$

1	1	1	2	1	1	3	1	1
1	1	2	2	1	2	3	1	2
1	1	3	2	1	3	3	1	3
1	2	1	2	2	1	3	2	1
1	2	2	2	2	2	3	2	2
1	2	3	2	2	3	3	2	3
1	3	1	2	3	1	3	3	1
1	3	2	2	3	2	3	3	2
1	3	3	2	3	3	3	3	3

$M_3 = 27$

1	1	1	1	2	1	1	1	3	1	1	1	4	1	1	1
1	1	1	2	2	1	1	2	3	1	1	2	4	1	1	2
1	1	1	3	2	1	1	3	3	1	1	3	4	1	1	3
1	1	1	4	2	1	1	4	3	1	1	4	4	1	1	4
1	1	2	1	2	1	2	1	3	1	2	1	4	1	2	1
1	1	2	2	2	1	2	2	3	1	2	2	4	1	2	2
1	1	2	3	2	1	2	3	3	1	2	3	4	1	2	3
1	1	2	4	2	1	2	4	3	1	2	4	4	1	2	4
1	1	3	1	2	1	3	1	3	1	3	1	4	1	3	1
1	1	3	2	2	1	3	2	3	1	3	2	4	1	3	2
1	1	3	3	2	1	3	3	3	1	3	3	4	1	3	3
1	1	3	4	2	1	3	4	3	1	3	4	4	1	3	4
1	1	4	1	2	1	4	1	3	1	4	1	4	1	4	1
1	1	4	2	2	1	4	2	3	1	4	2	4	1	4	2
1	1	4	3	2	1	4	3	3	1	4	3	4	1	4	3
1	1	4	4	2	1	4	4	3	1	4	4	4	1	4	4
1	2	1	1	2	2	1	1	3	2	1	1	4	2	1	1
...				...				...				...			

$M_4 = 64$

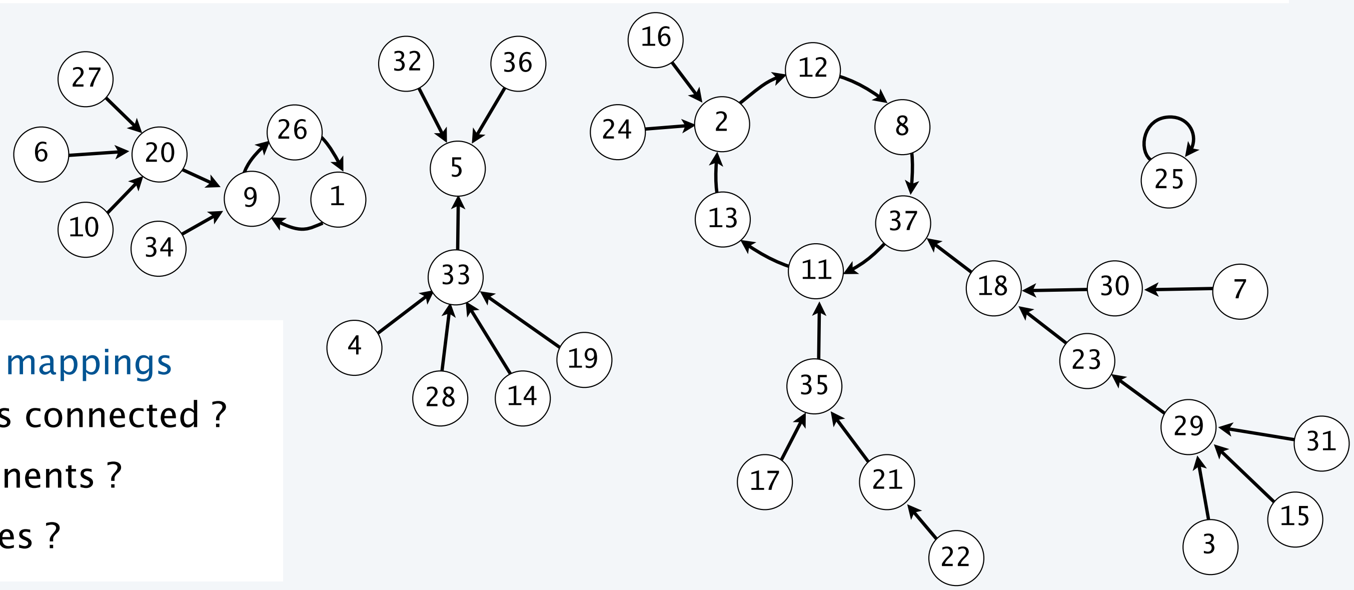
A. N^N

Digraph model for mappings

Every mapping corresponds to a digraph.

- *N* vertices, *N* edges.
- Every node has outdegree 1.
- Every node has indegree between 0 and *N*.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
9	12	29	33	5	20	30	37	26	20	13	8	2	33	29	2	35	37	33	9	35	21	18	2	25	1	20	33	23	18	29	5	5	9	11	5	11

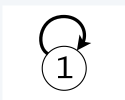


Natural questions about random mappings

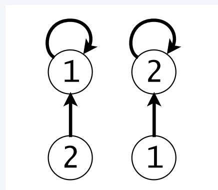
- Probability that the digraph is connected ?
- How many connected components ?
- How many nodes are on cycles ?

Cayley trees

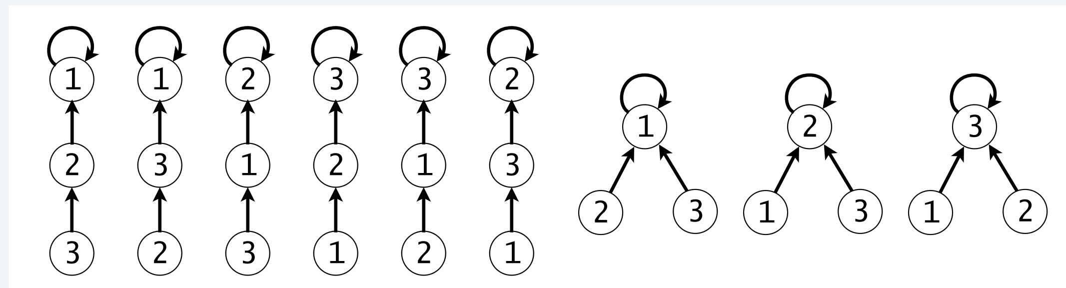
Q. How many different labeled rooted unordered trees of size N?



$C_1 = 1$



$C_2 = 2$

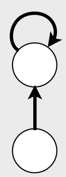


$C_3 = 9$

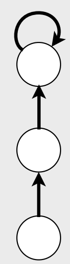
Short form: N-words grouped with unlabeled trees

1

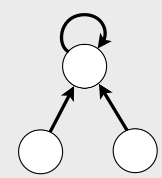
1 1
2 2



1 1 2
1 3 1
2 2 1
2 3 3
3 1 3
3 2 2



1 1 1
2 2 2
3 3 3



A. N^{N-1} via *Lagrange inversion* (see next slide)

Lagrange inversion

is a classic method for computing a *functional inverse*.

Def. The *inverse* of a function $f(u) = z$ is the function $u = g(z)$.

Ex. $f(u) = \dfrac{u}{1-u}$ $\quad g(z) = \dfrac{z}{1+z}$

Lagrange Inversion Theorem.

If a GF $g(z) = \sum_{n\geq 1} g_n z^n$ satisfies the equation $z = f(g(z))$ with $f(0) = 0$ and $f'(0) \neq 0$ then $g_n = \dfrac{1}{n}[u^{n-1}]\left(\dfrac{u}{f(u)}\right)^n$.

Proof. Omitted (best understood via complex analysis.

Ex. $f(u) = \dfrac{u}{1-u}$ $\qquad g_n = \dfrac{1}{n}[u^{n-1}](1-u)^n = (-1)^{n-1}$ $\qquad \sum_{n\geq 1}(-1)^n z^n = \dfrac{z}{1+z}$ ✓

Analytic combinatorics context: A widely applicable analytic transfer theorem

Lagrange-Bürmann inversion

A more general (and more useful) formuation:

Lagrange Inversion Theorem (Bürmann form).

If a GF $g(z) = \sum_{n\geq 1} g_n z^n$ satisfies the equation $z = f(g(z))$

with $f(0) = 0$ and $f'(0) \neq 0$ then, for any function $H(u)$,

$$[z^n]H(g(z)) = \frac{1}{n}[u^{n-1}]H'(u)\left(\frac{u}{f(u)}\right)^n$$

$H(u) = u$ gives the basic theorem

Stay tuned for applications.

Lagrange inversion: classic application

How many binary trees with N external nodes?

Class	T, the class of all binary trees
Size	The number of external nodes

Construction

$$T = Z + T \times T$$

OGF equation

$$T(z) = z + T(z)^2$$

$$z = T(z) - T(z)^2$$

Extract coefficients by Lagrange inversion with $f(u) = u - u^2$

$$[z^N]T(z) = \frac{1}{N}[u^{N-1}]\left(\frac{1}{1-u}\right)^N$$

$$= \frac{1}{N}\binom{2N-2}{N-1} \checkmark$$

Lagrange Inversion Theorem.

If a GF $g(z) = \sum_{n\geq 1} g_n z^n$ satisfies the equation $z = f(g(z))$ with $f(0) = 0$ and $f'(0) \neq 0$ then $g_n = \frac{1}{n}[u^{n-1}]\left(\frac{u}{f(u)}\right)^n$.

Take $M = N$ and $k = N - 1$ in

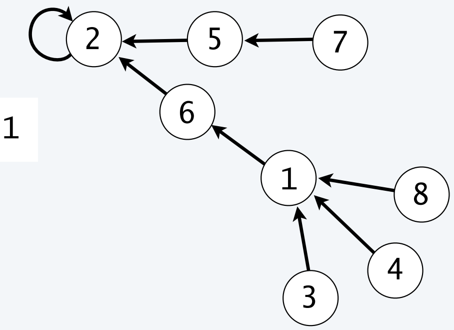
$$\frac{1}{(1-z)^M} = \sum_{k\geq 0}\binom{k+M-1}{M-1}z^k$$

Cayley trees

Class	$\mathcal{C}$, the class of labeled rooted unordered trees
EGF	$C(z) = \sum_{c \in \mathcal{C}} \frac{z^{|c|}}{|c|!} \equiv \sum_{N \geq 0} C_N \frac{z^N}{N!}$

Example

6 2 1 1 2 2 5 1



Construction $\quad C = Z \star (SET(C))$ ⟵ "a tree is a root connected to a set of trees"

EGF equation $\quad C(z) = ze^{C(z)}$

Extract coefficients by Lagrange inversion with $f(u) = u/e^u$

$$[z^N]C(z) = \frac{1}{N}[u^{N-1}]\Big(\frac{u}{u/e^u}\Big)^N$$

$$= \frac{1}{N}[u^{N-1}]e^{uN} = \frac{N^{N-1}}{N!}$$

$$C_N = N![z^N]C(z) = N^{N-1}$$ ✓

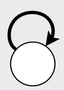
Lagrange Inversion Theorem.

If a GF $g(z) = \sum_{n \geq 1} g_n z^n$ satisfies the equation $z = f(g(z))$ with $f(0) = 0$ and $f'(0) \neq 0$ then $g_n = \frac{1}{n}[u^{n-1}]\Big(\frac{u}{f(u)}\Big)^n$.

Connected components in mappings

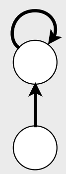
Q. How many different cycles of Cayley trees of size N ?

1

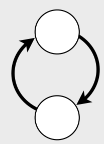


$Y_1 = 1$

1	1
2	2

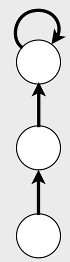


2	1

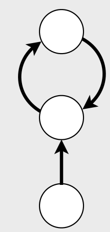


$Y_2 = 3$

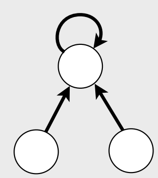
1	1	2
1	3	1
2	2	1
2	3	3
3	1	3
3	2	2



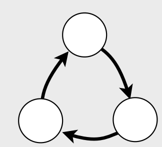
2	1	1
2	1	2
2	3	2
3	1	1
3	3	1
3	3	2



1	1	1
2	2	2
3	3	3



2	3	1
3	1	2



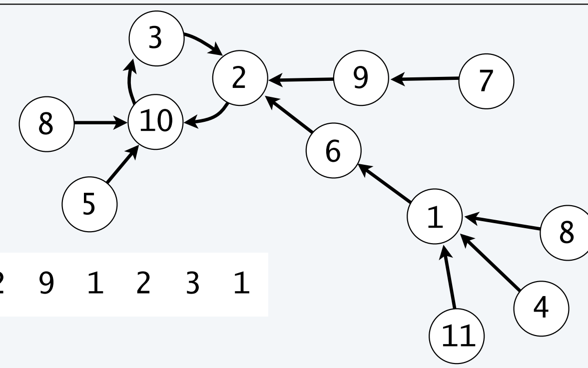
$Y_3 = 17$

A. $\sim \dfrac{N^N\sqrt{\pi}}{\sqrt{2N}}$ (see next slide)

Connected components in mappings

Class	Y, the class of cycles of Cayley trees
EGF	$Y(z) = \sum_{y \in Y} \frac{z^{\lvert y \rvert}}{\lvert y \rvert !} \equiv \sum_{N \geq 0} Y_N \frac{z^N}{N!}$

Example



1 10 2 1 10 2 9 1 2 3 1

Construction $\qquad Y = CYC(C) \longleftarrow$ "a component is a cycle of trees"

EGF equation

$$Y(z) = \ln \frac{1}{1 - C(z)}$$

Lagrange Inversion Theorem (Bürmann form).

If a GF $g(z) = \sum_{n \geq 1} g_n z^n$ satisfies the equation $z = f(g(z))$ with $f(0) = 0$ and $f'(0) \neq 0$ then, for any function $H(u)$,

$$[z^n]H(g(z)) = \frac{1}{n}[u^{n-1}]H'(u)\left(\frac{u}{f(u)}\right)^n$$

Extract coefficients by Lagrange inversion with $f(u) = u/e^u$ and $H(u) = \ln(1/(1-u))$

$$[z^N]Y(z) = \frac{1}{N}[u^{N-1}]\frac{1}{1-u}e^{uN}$$

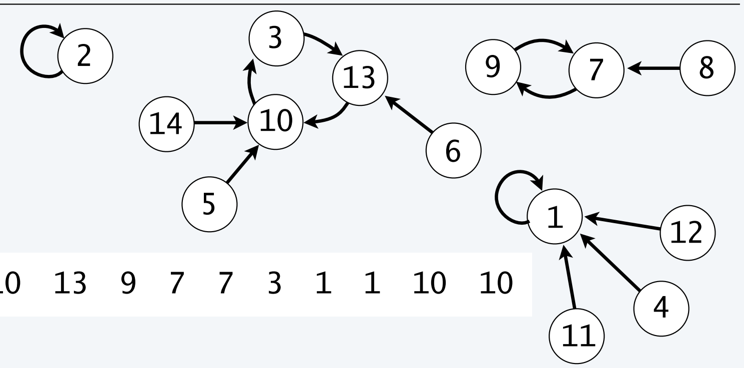
$$= \sum_{0 \leq k < N} \frac{N^{k-1}}{k!} = \sum_{1 \leq k \leq N} \frac{N^{N-k-1}}{(N-k)!}$$

$$Y_N = N![z^N]Y(z) = N^{N-1} \sum_{1 \leq k \leq N} \frac{N!}{N^k (N-k)!} = N^{N-1} Q(N) \sim \frac{N^N \sqrt{\pi}}{\sqrt{2N}}$$

Mappings

Class	M, the class of mappings
EGF	$M(z) = \sum_{m \in M} \frac{z^{\lvert m \rvert}}{\lvert m \rvert!} \equiv \sum_{N \geq 0} M_N \frac{z^N}{N!}$

Example



1 2 13 1 10 13 9 7 7 3 1 1 10 10

Construction

$$M = SET(CYC(C))$$

"a mapping is a set of cycles of trees"

EGF equation

$$M(z) = \exp\left(\ln \frac{1}{1 - C(z)}\right) = \frac{1}{1 - C(z)}$$

Lagrange Inversion Theorem (Bürmann form).

If a GF $g(z) = \sum_{n \geq 1} g_n z^n$ satisfies the equation $z = f(g(z))$ with $f(0) = 0$ and $f'(0) \neq 0$ then, for any function $H(u)$,

$$[z^n]H(g(z)) = \frac{1}{n}[u^{n-1}]H'(u)\left(\frac{u}{f(u)}\right)^n$$

Extract coefficients by Lagrange-Bürmann with $f(u) = u/e^u$ and $H(u) = 1/(1-u)$

$$[z^N]M(z) = \frac{1}{N}[u^{N-1}]\frac{1}{(1-u)^2}e^{uN}$$

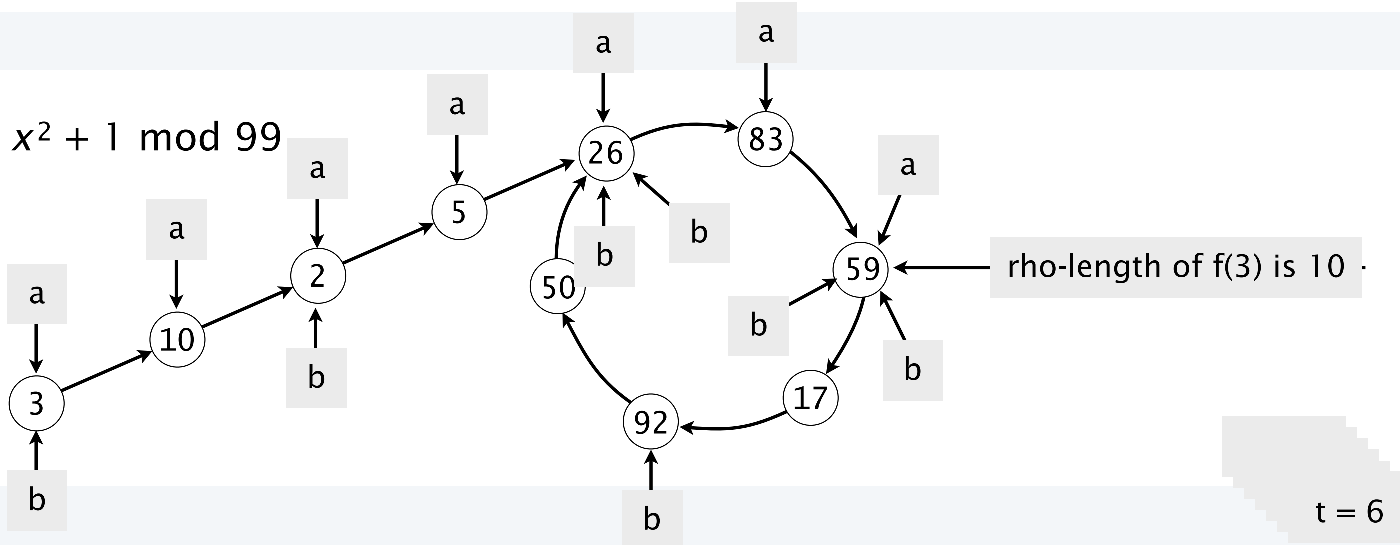
$$= \sum_{0 \leq k \leq N} (N-k)\frac{N^{k-1}}{k!} = \sum_{0 \leq k \leq N} \frac{N^k}{k!} - \sum_{1 \leq k \leq N} \frac{N^{k-1}}{(k-1)!} = \frac{N^N}{N!}$$

$$M_N = N^N \checkmark$$

Rho length

Def. The *rho-length* of a function at a given point is the number of iterates until it repeats.

Ex. $f(x) = x^2 + 1 \bmod 99$



Q. Algorithm to compute rho length ?

A. Symbol table? NO, rho length may be huge.

A. Floyd's "*tortoise-and-hare*" algorithm

Floyd's algorithm

```
int a = x, b = f(x), t = 0;
while (a != b)
{ a = f(a); b = f(f(b)); t++; }
// rho-length of f(a) is between t and 2t
```

Mapping parameters

are available via EBGFs based on the same constructions

Ex 1. Number of components

Construction $M = SET(uCYC(C))$

EGF equation $M(z) = \exp\left(u \ln \frac{1}{1 - C(z)}\right) = \frac{1}{(1 - C(z))^u}$

Ex 2. Number of trees (nodes on cycles)

Construction $M = SET(CYC(uC))$

EGF equation $M(z) = \exp\left(\ln \frac{1}{1 - uC(z)}\right) = \frac{1}{1 - uC(z)}$

Q. # of components?	A. $\ln \sqrt{N}$
Q. # of trees?	A. $\sqrt{\pi N}$
Q. Tail length?	A. $\sqrt{\pi N/8}$
Q. Rho length?	A. $\sim \sqrt{\pi N/2}$

Stay tuned to Part II for asymptotics.

Application: Pollard's rho-method for factoring

factors an integer N by iterating a random quadratic function to find a cycle.

Q. How does it work ?

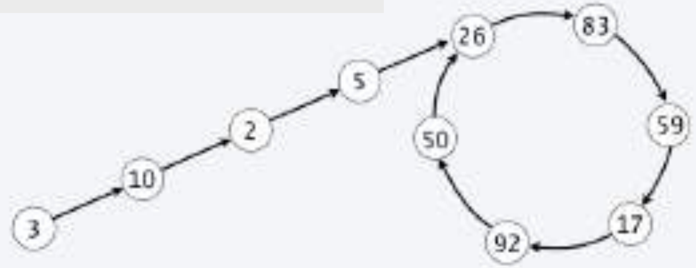
A. Iterate $f(x) = x^2 + c$ until finding a cycle ala Floyd's algorithm. Use a random value of c and start at a random point.

Pollard's algorithm

```
long a = (long) (Math.random()*N), b = a;
long c = (long) (Math.random()*N), d = 1;
while (d == 1)
{
   a = (a*a + c) % N;
   b = (b*b + c)*(b*b + c) + c % N;
   if (a > b) d = gcd((a - b) % N, N);
   else       d = gcd((b - a) % N, N);
}
// d is a factor of N.
```

need arbitrary-precision integer arithmetic package in real life

Ex. $N = 99$ (with $c = 1$)



a	3	10	2	5	
b	3	2	26	59	
d	1	1	1	3	✓

Application: Pollard's rho-method for factoring

factors an integer N by iterating a random quadratic function to find a cycle.

Q. How does it work ?

A. Iterate $f(x) = x^2 + c$ until finding a cycle ala Floyd's algorithm. Use a random value of c and start at a random point.

Q. Why does it work ?

A. Easy if you know number theory.

"magic" if you don't

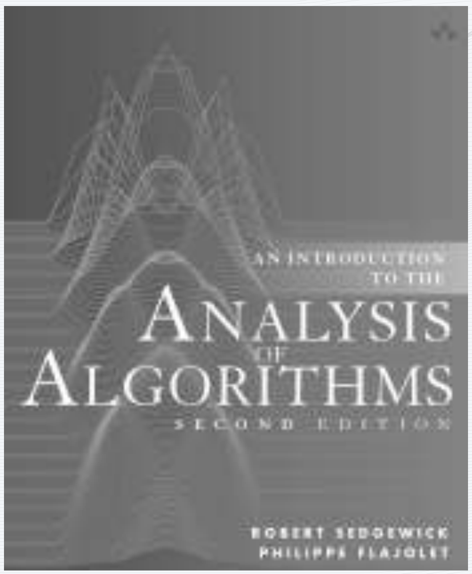
Pollard's algorithm

```
long a = (long) (Math.random()*N), b = a;
long c = (long) (Math.random()*N), d = 1;
while (d == 1)
{
   a = (a*a + c) % N;
   b = (b*b + c)*(b*b + c) + c % N;
   if (a > b) d = gcd((a - b) % N, N);
   else       d = gcd((b - a) % N, N);
}
// d is a factor of N.
```

Q. How many iterations ?

A. $\sim \sqrt{\pi N/2}$ *if* random quadratic functions are asymptotically equivalent to random mappings.

conjectured to be true but still open



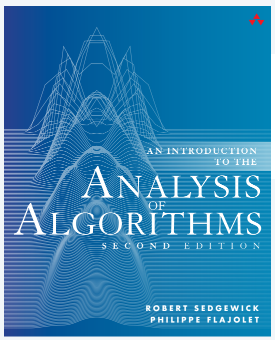
http://aofa.cs.princeton.edu

9. Words and Mappings

- Words
- Birthday problem
- Coupon collector problem
- Hash tables
- Mappings
- **Exercises**

Exercise 9.5

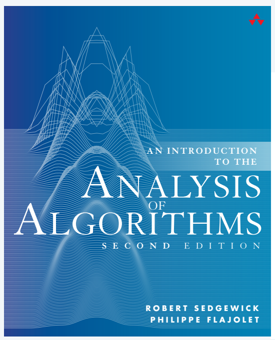
Being *really* sure that the birthday trick will work.



Exercise 9.5 For $M = 365$, how many people are needed to be 99% sure that two have the same birthday?

Exercise 9.38

Abel's binomial theorem (easier than it looks).

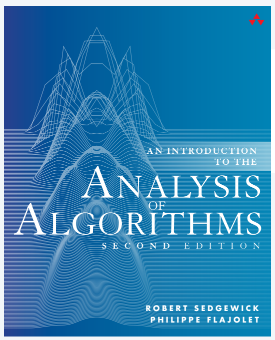


Exercise 9.38 ("Abel's binomial theorem.") Use the result of the previous exercise and the identity $e^{(\alpha+\beta)C(z)} = e^{\alpha C(z)} e^{\beta C(z)}$ to prove that

$$(\alpha+\beta)(n+\alpha+\beta)^{n-1} = \alpha\beta \sum_k \binom{n}{k} (k+\alpha)^{k-1} (n-k+\beta)^{n-k-1}.$$

Exercise 9.99

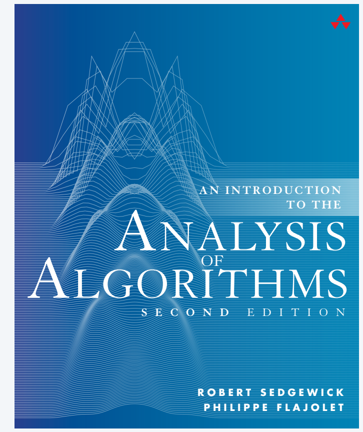
[Not in the book, but should be there.]



Exercise 9.99 Show that the probability that a random mapping of size N has no singleton cycles is $\sim N/e$, the same as for permutations (!).

Assignments

1. Read pages 473-542 in text.

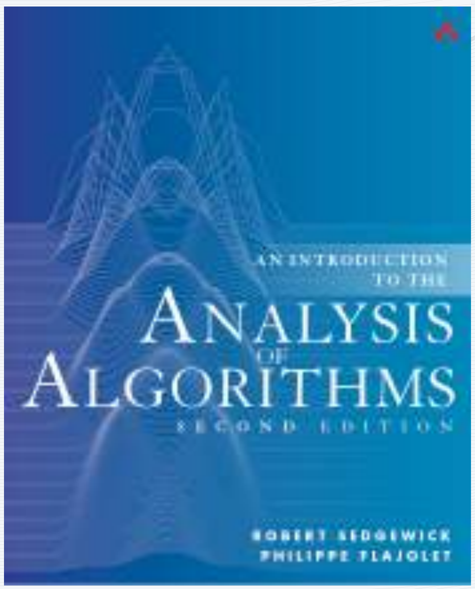


2. Run experiments to validate mathematical results.

Experiment 9.1. Implement linear probing hashing and run experiments for $N = 1$ million and $M = 900{,}000$ to validate the prediction from Knuth's analysis that about 5.5 probes should be needed, on average, for a successful search.

Experiment 9.2. [Exercise 9.51] Write a program to find the rho length and tree path length of a random mapping. Generate 1000 random mappings for N as large as you can and compute the average number of cycles, rho length, and tree path length.

3. Write up solutions to Exercises 9.5, 9.38, and 9.99.



http://aofa.cs.princeton.edu

9. Words and Mappings