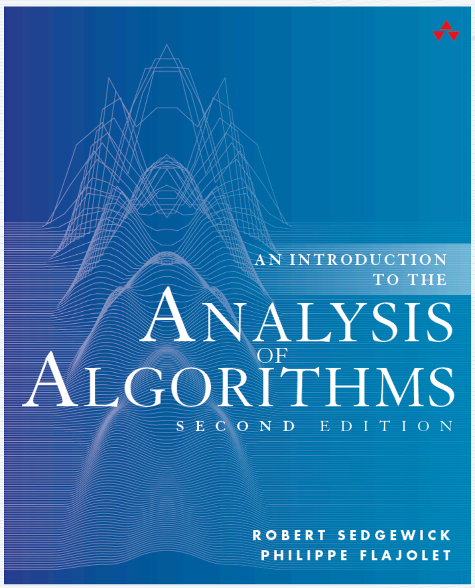
ANALYTIC COMBINATORICS

PART ONE



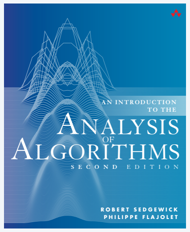
http://aofa.cs.princeton.edu

9. Words and Mappings

Orientation

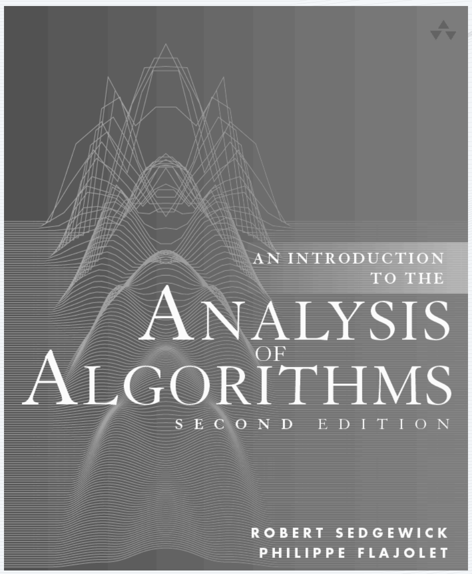
Second half of class

- Surveys fundamental combinatorial classes.
- Considers techniques from analytic combinatorics to study them .
- Includes applications to the analysis of algorithms.



chapter	*combinatorial classes*	*type of class*	*type of GF*
6	Trees	unlabeled	OGFs
7	Permutations	labeled	EGFs
8	Strings and Tries	unlabeled	OGFs
9	Words and Mappings	labeled	EGFs

Note: Many more examples in book than in lectures.



http://aofa.cs.princeton.edu

9. Words and Mappings

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

Symbolic method for unlabelled objects (review)

Warmup: How many binary strings with N bits?

Class	B, the class of all binary strings
Size	$\|b\|$, the number of bits in b
OGF	$B(z) = \sum_{b \in B} z^{\|b\|} = \sum_{N \geq 0} B_N z^N$

Atoms

type	*class*	*size*	*GF*
0 bit	Z_0	1	z
1 bit	Z_1	1	z

Construction $\quad B = SEQ(Z_0 + Z_1)$

"a binary string is a sequence of 0 bits and 1 bits"

OGF equation $\quad B(z) = \dfrac{1}{1 - 2z}$

$$[z^N]B(z) = 2^N \quad \checkmark$$

Symbolic method for unlabelled objects (review)

How many strings drawn from an *M*-char alphabet with *N* chars?

Class	S, the class of all strings
Size	$\lvert s \rvert$, the number of chars in s
OGF	$S(z) = \sum_{s \in S} z^{\lvert s \rvert} = \sum_{N \geq 0} S_N z^N$

Atoms

type	*class*	*size*	*GF*
char 1	Z_1	1	z
char 2	Z_2	1	z
. . .			
char M	Z_M	1	z

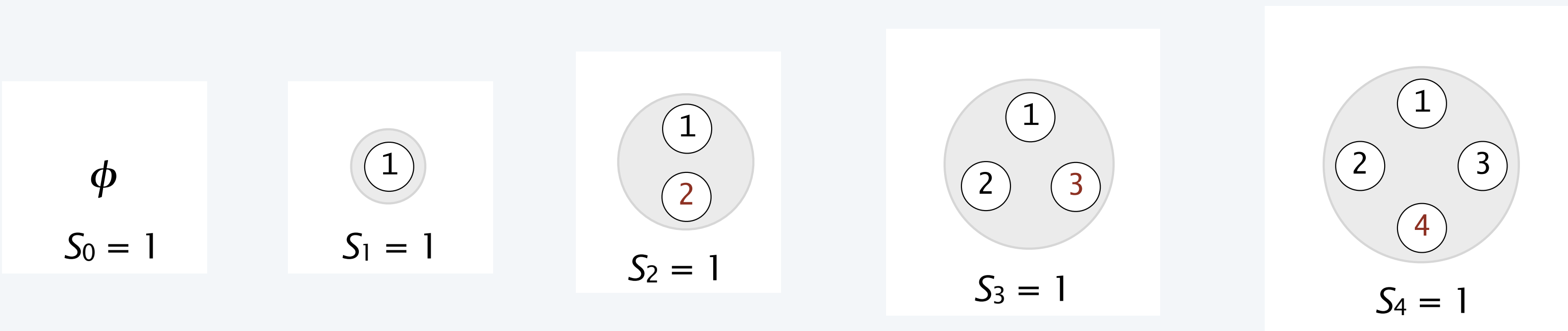
Construction $S = SEQ(Z_1 + Z_2 + \ldots + Z_M)$ "a string is a sequence of chars"

OGF equation $$S(z) = \frac{1}{1 - Mz}$$

Extract coefficients $[z^N]S(z) = M^N$ ✓

Symbolic method for labelled objects (review): sets

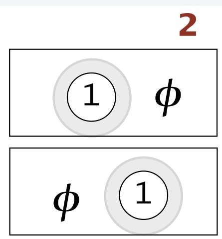
Q. How many labeled sets (urns) of size N?



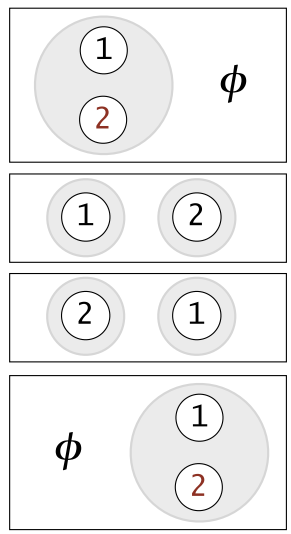
A. One.

Labelled objects review (continued): sets

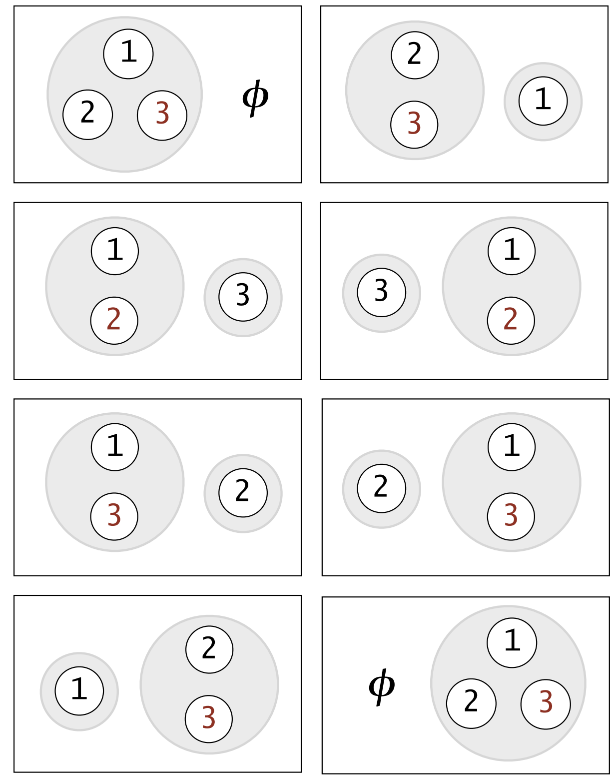
Q. How many ordered pairs of labelled sets of N objects?



$P_1 = 2$



$P_2 = 4$



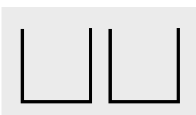
$P_3 = 8$

A. 2^N

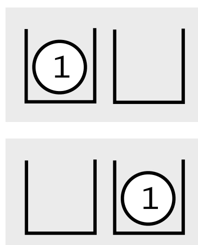
Q. How many sequences of length M of urns with N objects in total ?

Balls-and-urns viewpoint

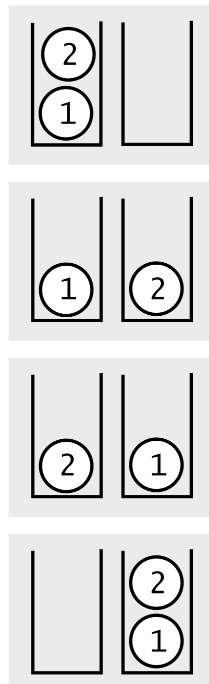
Q. How many different ways to throw N balls into 2 urns?



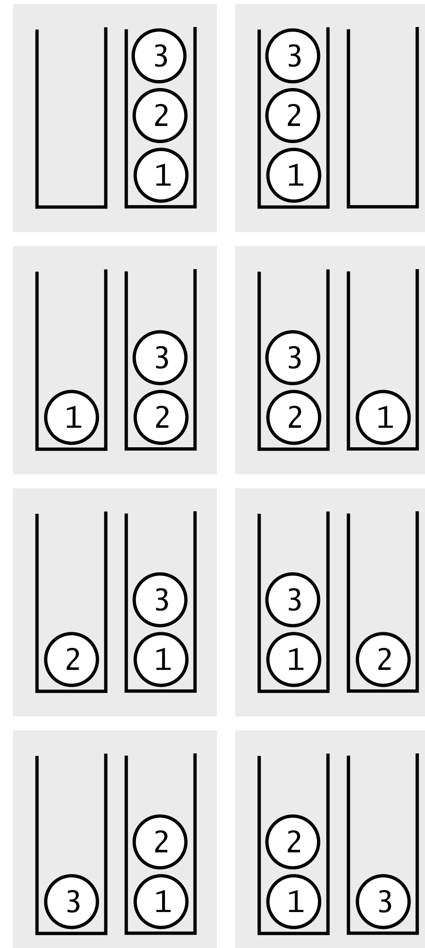
$W_0 = 1$



$W_1 = 2$



$W_2 = 4$



$W_3 = 8$

A. 2^N

The symbolic method for labelled classes (review)

Theorem. Let A and B be combinatorial classes of labelled objects with EGFs $A(z)$ and $B(z)$. Then

construction	*notation*	*semantics*	*EGF*
disjoint union	$A + B$	disjoint copies of objects from A and B	$A(z) + B(z)$
labelled product	$A \star B$	ordered pairs of copies of objects, one from A and one from B	$A(z)B(z)$
sequence	$SEQ_k(A)$	k- sequences of objects from A	$A(z)^k$
	$SEQ(A)$	sequences of objects from A	$\dfrac{1}{1 - A(z)}$
set	$SET_k(A)$	k-sets of objects from A	$A(z)^k/k!$
	$SET(A)$	sets of objects from A	$e^{A(z)}$
cycle	$CYC_k(A)$	k-cycles of objects from A	$A(z)^k/k$
	$CYC(A)$	cycles of objects from A	$\ln \dfrac{1}{1 - A(z)}$

Words

Def. A *word* is a sequence of M urns holding N objects in total.

"throw N balls into M urns"

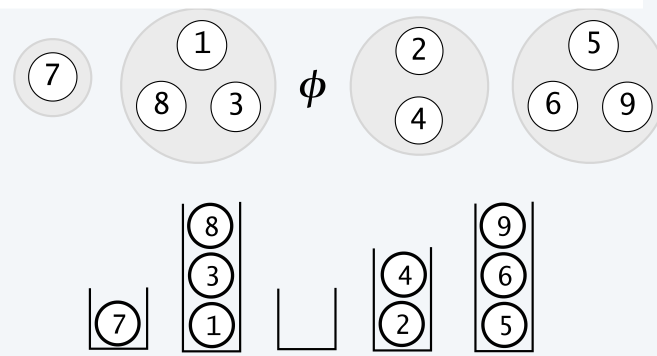
Q. How many words ?

Class	W_M, the class of M-sequences of urns
Size	$\|w\|$, the number of objects in w
EGF	$W_M(z) = \sum_{w \in W_M} \frac{z^{\|w\|}}{\|w\|!} = \sum_{N \geq 0} W_{MN} \frac{z^N}{N!}$

Atom

type	*class*	*size*	*GF*
labelled atom	Z	1	z

Example { 7 } { 1 8 3 } { } { 2 4 } { 5 6 9 }



Construction $\quad W_M = SEQ_M(SET(Z))$

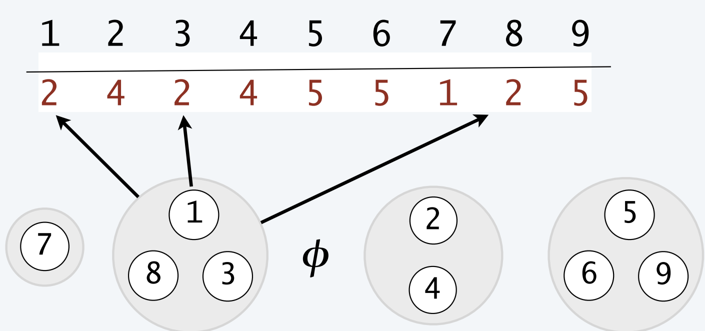
OGF equation $\quad W_M(z) = (e^z)^M = e^{Mz}$

Counting sequence $\quad N![z^N]W_M(z) = M^N$

A 1:1 correspondence

A string is a sequence of N characters (from an M-char alphabet). There are M^N strings.

A word is a sequence of M labelled sets (having N objects in total). There are M^N words.



Typical string

2 4 2 4 5 5 1 2 5

Typical word

{ 7 } { 1 8 3 } { } { 2 4 } { 5 6 9 }

Correspondence

- For each i in the kth set in the word set the ith char in the string to k.
- If the ith char in the string is k, put i into the kth set in the word.

Strings and Words

Familiar definition.
A string is a sequence of N characters (from an M-char alphabet).

Combinatorial definition.
A word is a sequence of M labeled sets (having N objects in total).

1-1 correspondence between words and strings
- Length of sequence in word: number of chars M in the alphabet.
- Number of objects in the set: length of string N.
- k th set in the sequence: indices where k appears in the string.

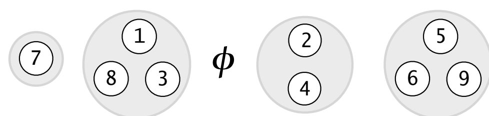
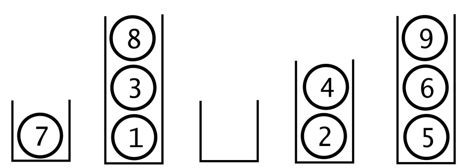
Q. What is the difference between strings and words?

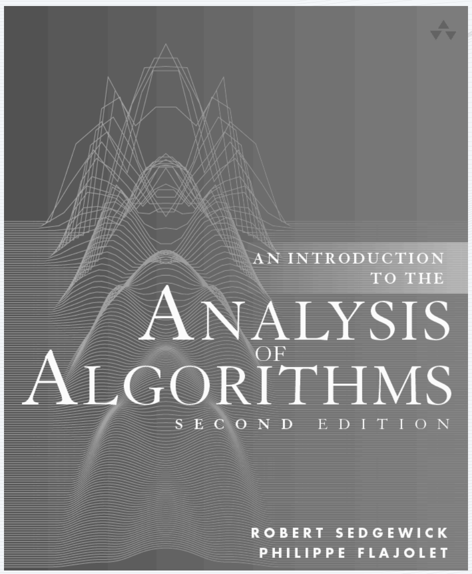
A. Only the point of view.
- With strings (last lecture) we study the sequence of characters.
- With words (this lecture) we study the sets of indices.

$N = 3$
$M = 2$

1 2 3	
0 0 0	{ 1 2 3 } { }
0 0 1	{ 1 2 } { 3 }
0 1 0	{ 1 3 } { 2 }
0 1 1	{ 1 } { 2 3 }
1 0 0	{ 2 3 } { 1 }
1 0 1	{ 2 } { 1 3 }
1 1 0	{ 3 } { 1 2 }
1 1 1	{ } { 1 2 3 }

Strings and Words (summary)

class	*type*	*GF type*	*typical*	*construction*	*GF*	*count*
STRING	unlabelled	OGF	2 4 2 4 5 5 1 2 5	$S = \text{SEQ}(Z_1 + \ldots + Z_M)$	$S(z) = \dfrac{1}{1 - Mz}$	M^N
WORD	labelled	EGF	 { 7 } { 1 8 3 } { } { 2 4 } { 5 6 9 } 2 4 2 4 5 5 1 2 5 	$W_M = \text{SEQ}_M(\text{SET}(Z))$	$W_M(z) = e^{Mz}$	M^N



http://aofa.cs.princeton.edu

9. Words and Mappings

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

Birthday problem

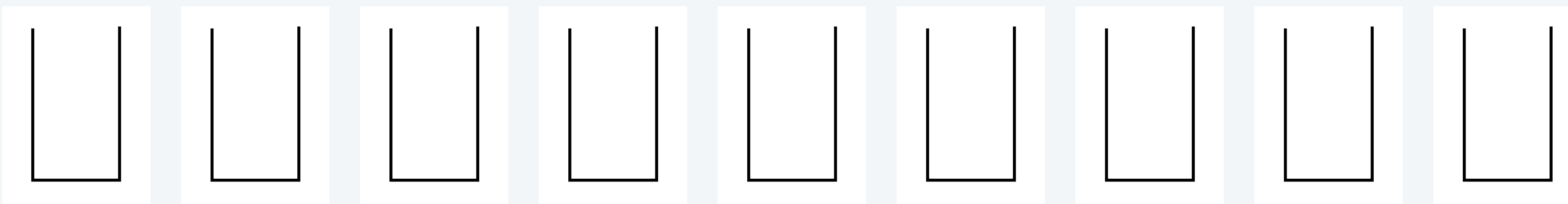
One at a time, ask each member of a group of people their birth date.

Q. How many people asked before finding two with the same birthday?

Quick answer: at most 365

Birthday problem

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



Q. How long until some urn gets two balls (for $M = 365$) ?

Birthday sequences (words with no duplicates)

Def. A *birthday sequence* is a word where no set has more than one element.

a string with no duplicate letters

Q. How many birthday sequences?

Class	B_M, the class of birthday sequences
EGF	$B_M(z) = \sum_{w \in B_M} \frac{z^{\lvert w \rvert}}{\lvert w \rvert !} = \sum_{N \geq 0} B_{MN} \frac{z^N}{N!}$

Example

{ 3 } { } { 5 } { 1 } { } { } { 4 } { 2 } { }

4 8 1 7 3

Construction $B_M = SEQ_M(E + Z)$

OGF equation $B_M(z) = (1 + z)^M$

Counting sequence

$$N![z^N]B_M(z) = N!\binom{M}{N} = \frac{M!}{(M-N)!}$$

$$= M(M-1)\ldots(M-N+1)$$

Birthday problem

Number of N-char M-words where no char is repeated

$$M(M-1)(M-2)\ldots(M-N+1) = \frac{M!}{(M-N)!}$$

Probability that no char is repeated in a random M-word of length N.

$$\frac{M!}{M^N(M-N)!}$$

Same as the probability that the first repeat position is > N.

Expected position of the first repeat

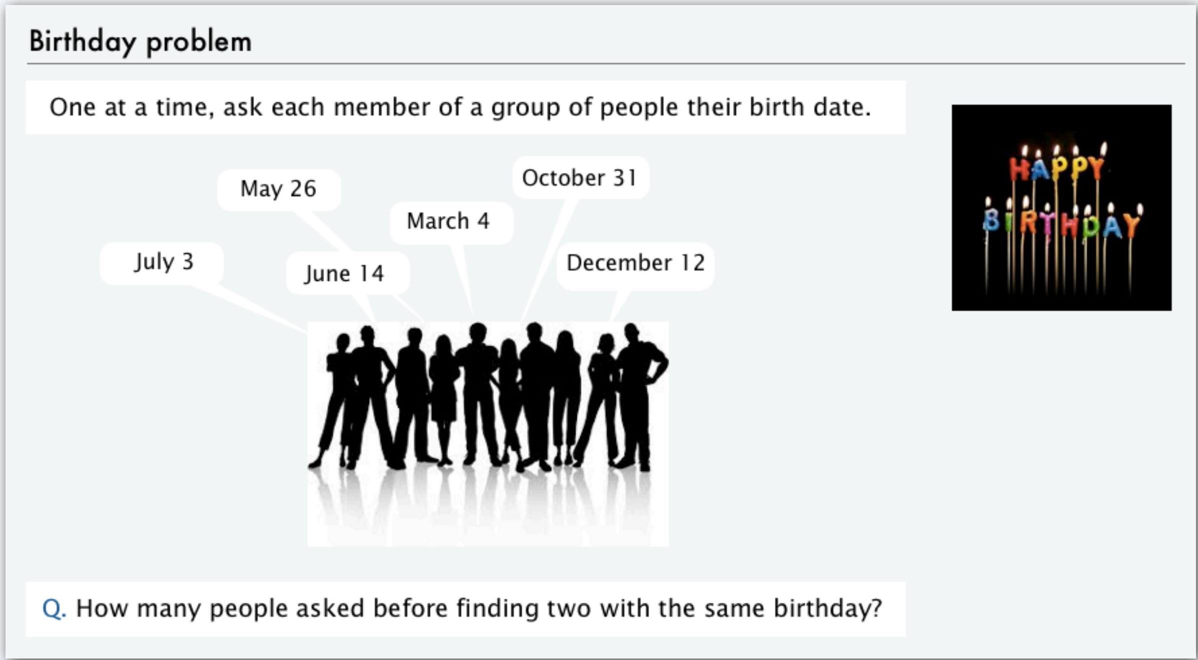
$$\sum_{0 \le N \le M} \frac{M!}{M^N(M-N)!}$$

Laplace method to estimate Ramanujan Q-function (see Asymptotics lecture)

$$= 1 + Q(M) \sim \sqrt{\pi M/2}$$

Theorem. Expected position of the first repeated character in a random M-word is $\sim \sqrt{\pi M/2}$

Birthday problem

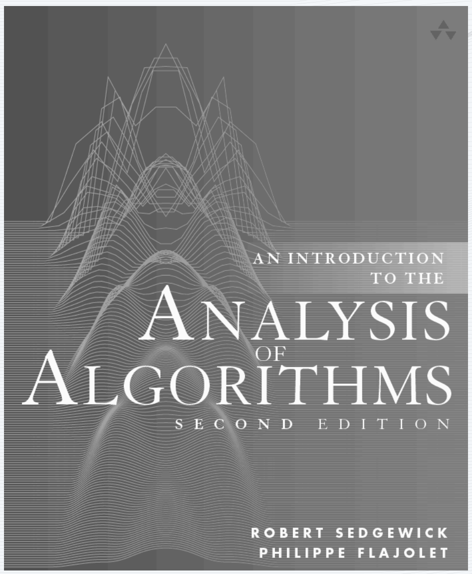


Q. How many people asked before finding two with the same birthday?

A. About 24.

```
% bc
scale = 5
sqrt(3.14159*365/2)
23.94453
```

$$\sim \sqrt{\pi M/2}$$



http://aofa.cs.princeton.edu

9. Words and Mappings

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

Coupon collector problem

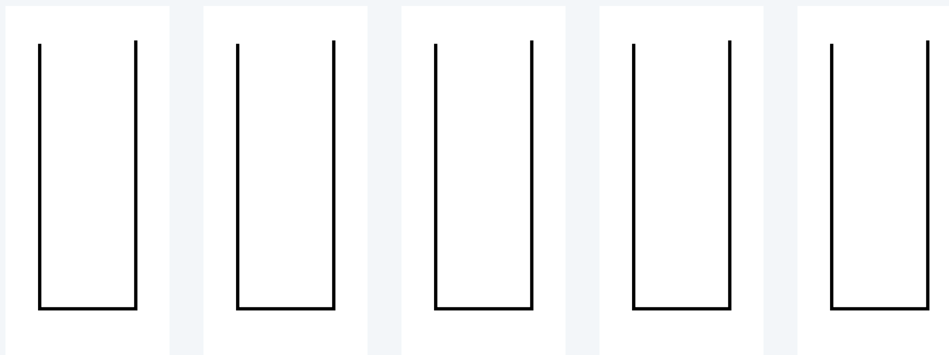
One at a time, ask each member of a group of people their birth date.

Q. How many people asked before finding *every day of the year*?

Quick answer: at *least* 365

Coupon collector problem

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



Q. How long until each urn has at least one ball ?

Coupon collector problem

A collector buys coupons, each randomly chosen from *M* different types

Q. How many coupons collected before having every possible coupon?

Quick answer: at *least* 365

Coupon collector problem

Roll an *M*-sided die.

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	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	14	5	5	9	4	5	6

first repeat

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Q. How many rolls until seeing all *M* values ?

Coupon collector (classical analysis)

Probability that more than j rolls are needed to get the (k+1)st coupon

$$\left(\frac{k}{M}\right)^j$$

Expected number of rolls to get the (k+1)st coupon

$$\sum_{j\geq 0}\left(\frac{k}{M}\right)^j = \frac{1}{1-k/M} = \frac{M}{M-k}$$

Expected number of rolls to get all coupons

$$\sum_{0\leq k<M}\frac{M}{M-k} = MH_M \sim M\ln M$$

by linearity of expectation

Theorem. Expected number of coupons needed to complete a collection of size M is $\sim M \ln M$.

Motivation for studying in more detail:

- Discover variance and other properties of the distribution.
- Learn structure suitable for analyzing variants and extensions.

Coupon collector sequences (M-words with no empty sets)

Def. A *coupon collector sequence* is an M-word with no empty set.

a string that uses all the letters in the alphabet

Q. How many coupon collector sequences?

Example ($M = 26$)

`the quick brown fox jumps over the lazy dog`

Example ($M = 5$)

2 4 2 4 5 5 1 5 3

{ 7 } { 1 3 } { 9 } { 2 4 } { 5 6 8 }

Class	R_M, the class of coupon collector sequences
EGF	$R_M(z) = \sum_{w \in R_M} \frac{z^{\vert w \vert}}{\vert w \vert !} = \sum_{N \geq 0} R_{MN} \frac{z^N}{N!}$

Construction $$R_M = SEQ_M(SET_{>0}(Z))$$

EGF equation $$R_M(z) = (e^z - 1)^M$$

Counting sequence $$N![z^N]R_M(z) = N![z^N]\sum_j \binom{M}{j}(-1)^j e^{(M-j)z}$$

$$= \sum_j \binom{M}{j}(-1)^j (M-j)^N \quad \sim M^N$$

Coupon collector sequences (EGF analysis, continued)

Probability that a random M-word of length N is a coupon collector sequence.

$$\frac{1}{M^N}\sum_j \binom{M}{j}(-1)^j(M-j)^N = \sum_j \binom{M}{j}(-1)^j\Big(1-\frac{j}{M}\Big)^N$$

Probability that collection in a random M-word completes in $>N$ chars.

$$1-\sum_j \binom{M}{j}(-1)^j\Big(1-\frac{j}{M}\Big)^N$$

Average number of chars to complete a collection in a random M-word.

$$\sum_{N\geq 0}\Big(1-\sum_j \binom{M}{j}(-1)^j\Big(1-\frac{j}{M}\Big)^N\Big)$$

$$= -M\sum_{j\geq 1}\binom{M}{j}\frac{(-1)^j}{j}$$

$$= MH_M$$

Knuth Exercise 1.2.7-13

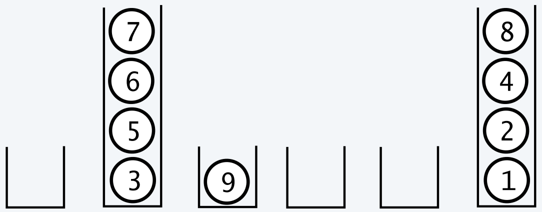
Coupon collector (OGF analysis)

Class	W_{Mk}, the class of *M*-words with k different letters and the last letter appearing only once
OGF	$W_{Mk}(z) = \sum_{w \in W_{Mk}} z^{\lvert w \rvert} = \sum_{N \geq 0} W_{MNk} z^N$
PGF	$W_{Mk}(z/M) = \sum_{N \geq 0} W_{MNk} \frac{z^N}{M^N}$
Mean wait time for k coupons	$w_{Mk} \equiv W'_{Mk}(z/M)\Big\rvert_{z=1} = \sum_{N \geq 0} N \frac{W_{MNk}}{M^N} z^N$

Example

6 6 2 6 2 2 2 6 3

{ } { 3 5 6 7 } { 9 } { } { } { 1 2 4 8 }



Coupon collector (OGF analysis, continued)

$W_{Mk} \equiv M$-words with k different letters and the last letter appearing only once.

Construction $\quad W_{Mk} = (k-1)Z \times W_{Mk} + (M-k+1)Z \times W_{M(k-1)}$

OGF equation $\quad (1-(k-1)z)W_{Mk}(z) = (M-(k-1))zW_{M(k-1)}(z)$

OGF
$W_{Mk}(z)$

Evaluate at z/M

$$(M-(k-1)z)W_{Mk}(z/M) = (M-(k-1))zW_{M(k-1)}(z/M)$$

PGF
$W_{Mk}(z/M)$

Differentiate and evaluate at 1

$$(M-(k-1))w_{Mk} - (k-1) = (M-(k-1))(w_{M(k-1)}+1)$$

Wait time for k coupons
$w_{Mk} \equiv W'_{Mk}(z/M)\Big|_{z=1}$

Rearrange terms and telescope

$$w_{Mk} = w_{M(k-1)} + \frac{k-1}{M-(k-1)} + 1 = w_{M(k-1)} + \frac{M}{M-(k-1)}$$

$$= \sum_{0 \le j < k} \frac{M}{M-j} = M(H_M - H_{M-k})$$

Wait time for full collection

$$w_{MM} = MH_M$$

Coupon collector problem

A collector buys coupons, each randomly chosen from M different types

Q. How many coupons collected before having every possible coupon?

A. $\sim M \ln M$.

Coupon collector problem

Roll an *M*-sided die.

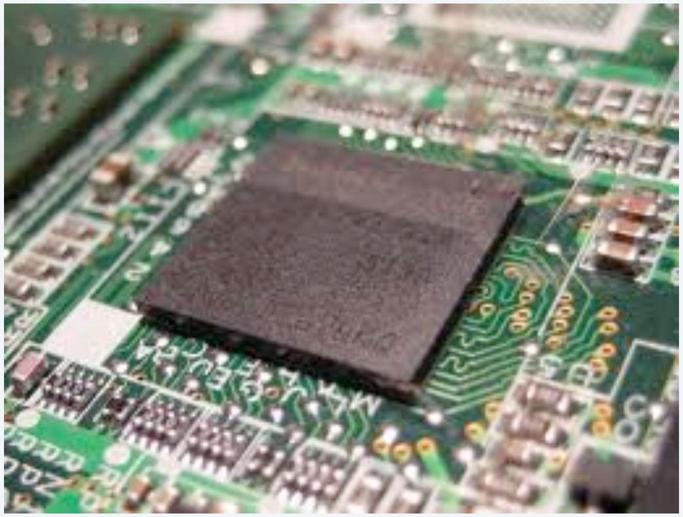
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
9	12	19	3	5	20	10	17	16	20	13	8	2	13	9	2	15	17	3	9	11	7	18	2	10	1	20	12	10	8	14	5	5	9	4	5	6

Q. How many rolls until seeing all M values?

A. $\sim M \ln M$. ← About 60 for a 20-sided die

Coupon collector problem: Sample application

A program randomly accesses an M-page memory.



Q. How many memory accesses before hitting every page, when $M = 2^{20}$?

A. About 14.5 million.

```
% bc -l
l(2)
.69314718055994530941
2^20*l(2^20)
14536349.96005650425534480384
```

Surjections

Def. An *M-surjection* is an *M*-word with no empty set.

Alt name for "coupon collector sequence"

Def. A *surjection* is a word that is an *M*-surjection for some *M*.

Q. How many surjections of length *N* ?

Class	R_M, the class of *M*-surjections

Construction

$$R_M = SEQ_M(SET_{>0}(Z))$$

EGF equation

$$R_M(z) = (e^z - 1)^M$$

Coefficients

$$R_{MN} \sim M^N$$

Class	R, the class of surjections

Construction

$$R = SEQ(SET_{>0}(Z))$$

EGF equation

$$R(z) = \frac{1}{1-(e^z-1)} = \frac{1}{2-e^z}$$

Coefficients

$$N![z^N]R(z) \sim \frac{N!}{2(\ln 2)^{N+1}}$$

Best handled with complex asymptotics (stay tuned for Part II)

1

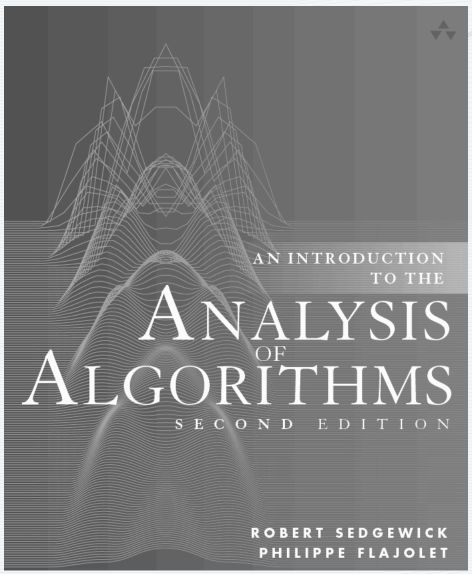
$R_1 = 1$

1 1
1 2
2 1

$R_2 = 3$

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

$R_3 = 13$



http://aofa.cs.princeton.edu

9. Words and Mappings

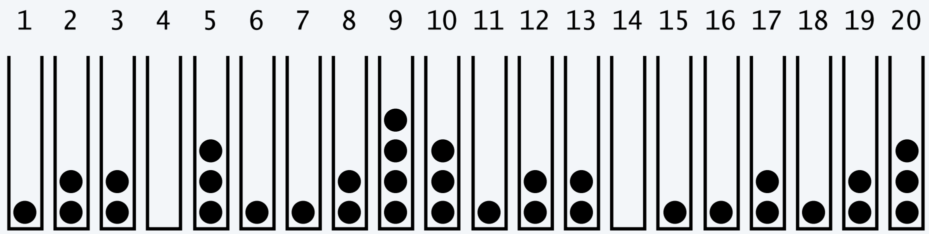
- Words
- Birthday problem
- Coupon collector problem
- **Hash tables**
- Mappings

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}

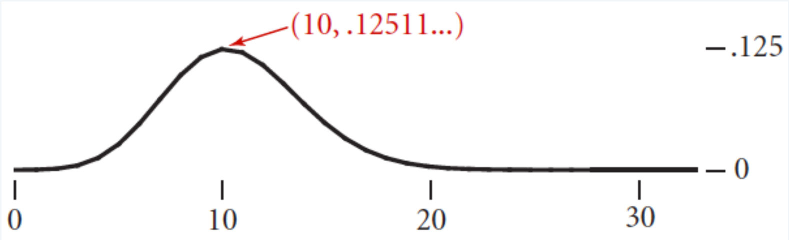
Theorem. The probability that a value occurs *k* times in a random *M*-word of length *N* is

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

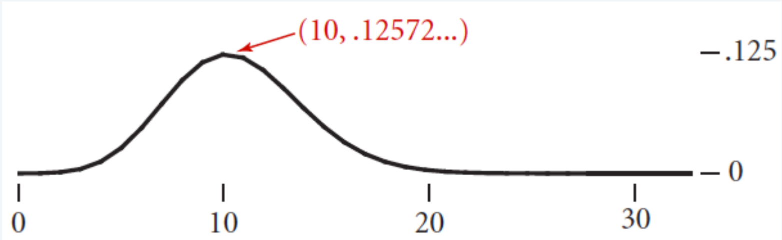
For $\alpha = N/M$ fixed and $k = O(1)$, this is

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

Proof. [See Lecture 4.]

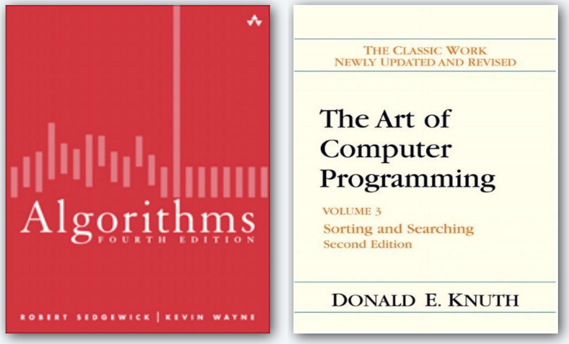


Binomial distribution for $N = 10^4$ and $M = 10^3$ ($\alpha = 10$).



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)

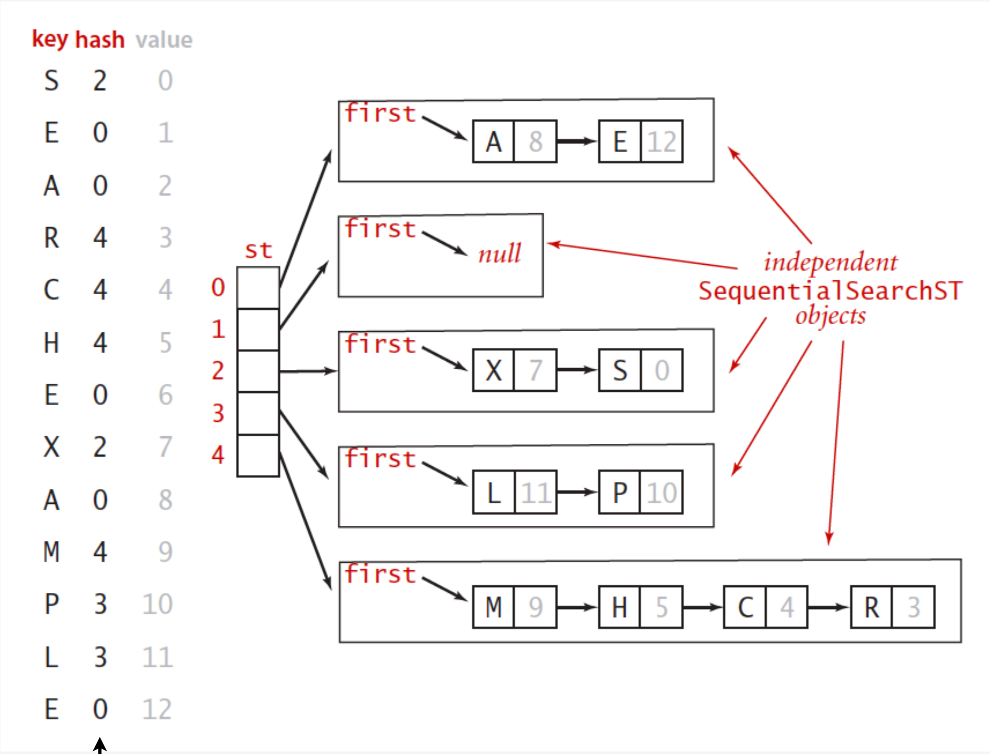
- *Separate chaining*—keep M linked lists, one for each hash value.
- *Linear probing*—use an array and scan for empty spots on collision.

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.

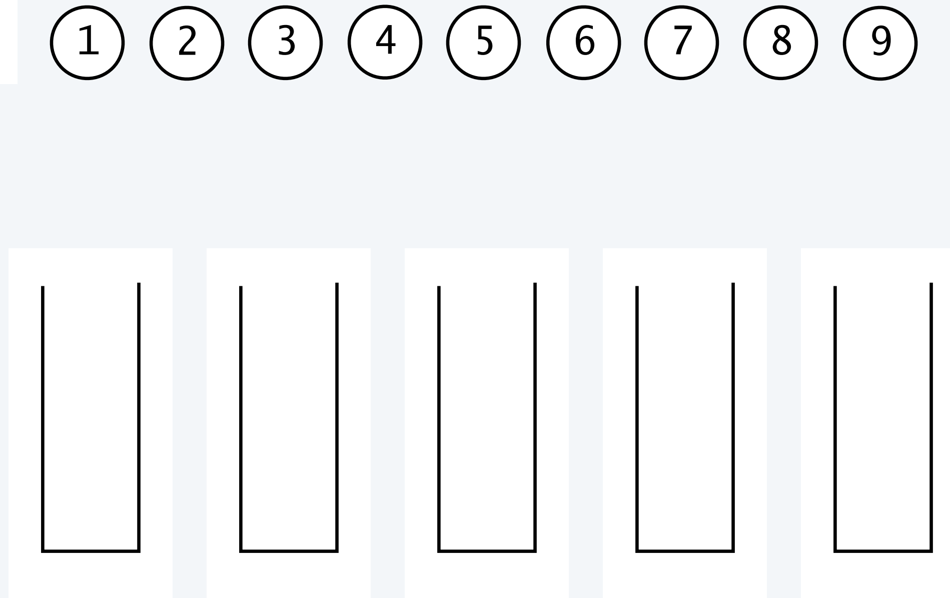


a random word!

Section 3.4

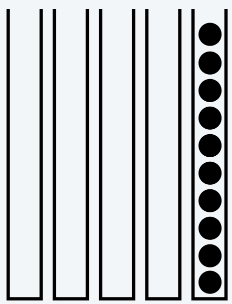
Application: Hashing with separate chaining

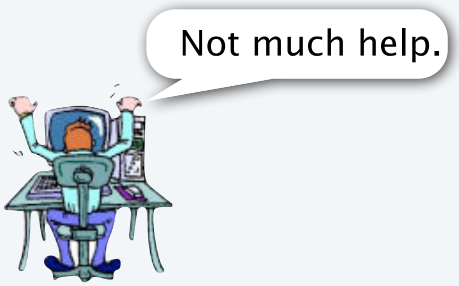
Throw N balls into M urns, one at a time.



Q. Average number of balls in each urn ?

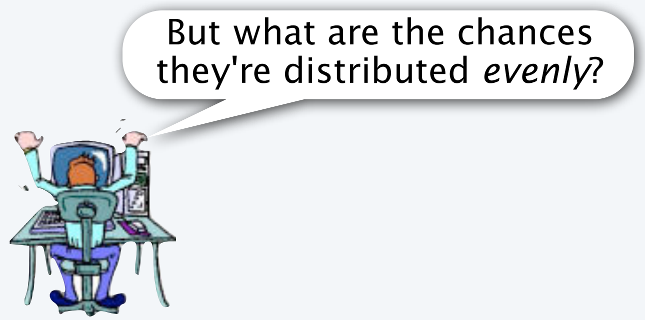
A. Obvious: N/M





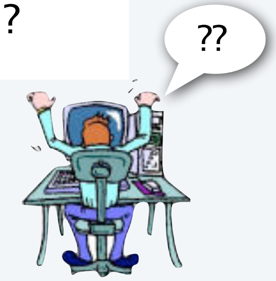
Q. Probability that a given urn has k balls ?

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

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

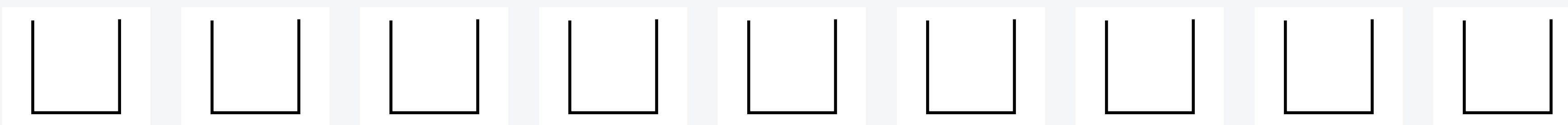
```
% bc - l
scale = 20
2*e(-10)*e(50)/5^50
.00000000000000000530
```

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.

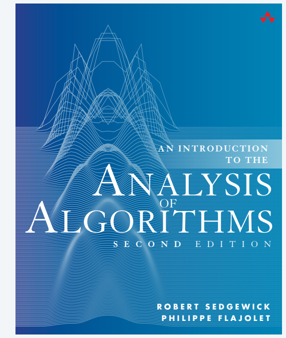
Q. Average number of probes to find one of N keys?

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

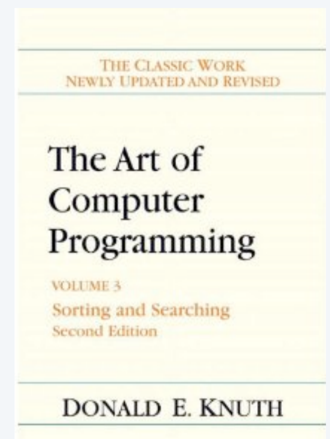
$$= Q(M) \sim \sqrt{\pi M/2}$$ when table is full ($N = M-1$)

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

Section 3.4



pp. 509—518



A footnote

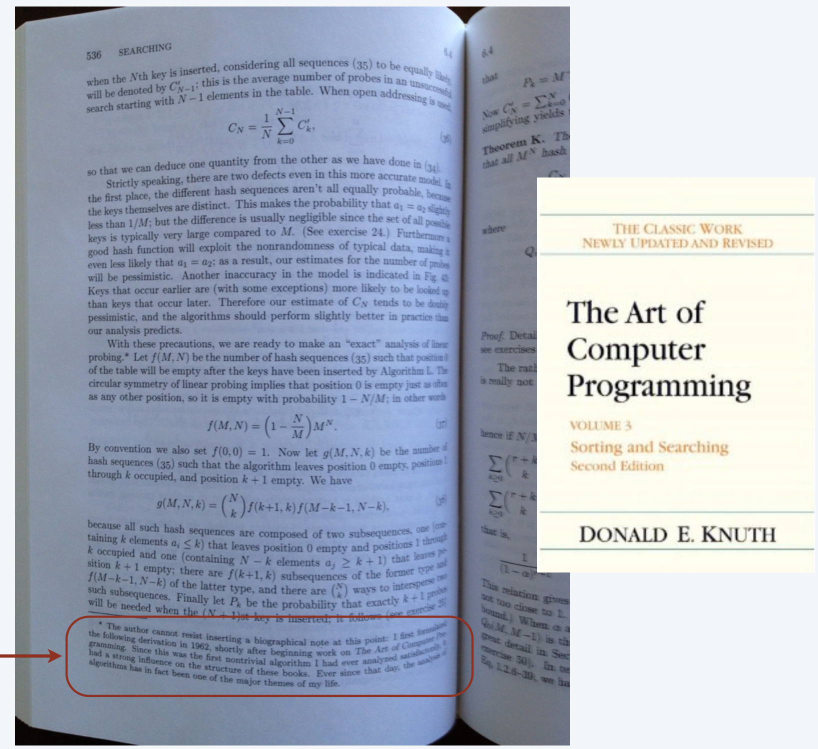
Q. Average number of probes to find one of N keys?

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



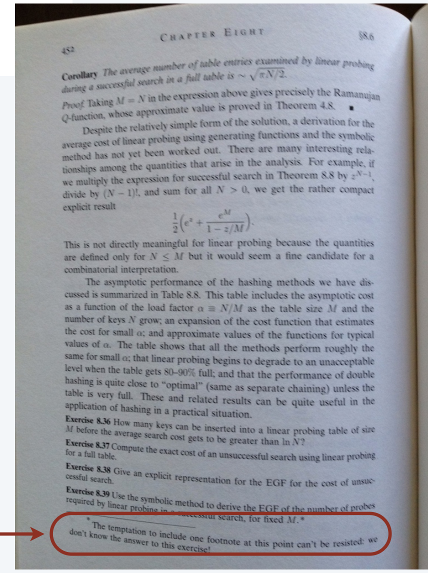
Another footnote

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



452 CHAPTER EIGHT §8.6

Corollary *The average number of table entries examined by linear probing during a successful search in a full table is* $\sim \sqrt{\pi N/2}$.

Proof. Taking $M = N$ in the expression above gives precisely the Ramanujan Q-function, whose approximate value is proved in Theorem 4.8. ∎

Despite the relatively simple form of the solution, a derivation for the average cost of linear probing using generating functions and the symbolic method has not yet been worked out. There are many interesting relationships among the quantities that arise in the analysis. For example, if we multiply the expression for successful search in Theorem 8.8 by z^{N-1}, divide by $(N-1)!$, and sum for all $N > 0$, we get the rather compact explicit result

$$\frac{1}{2}\left(e^z + \frac{e^M}{1 - z/M}\right).$$

This is not directly meaningful for linear probing because the quantities are defined only for $N \le M$ but it would seem a fine candidate for a combinatorial interpretation.

The asymptotic performance of the hashing methods we have discussed is summarized in Table 8.8. This table includes the asymptotic cost as a function of the load factor $\alpha \equiv N/M$ as the table size M and the number of keys N grow; an expansion of the cost function that estimates the cost for small α; and approximate values of the functions for typical values of α. The table shows that all the methods perform roughly the same for small α; that linear probing begins to degrade to an unacceptable level when the table gets 80–90% full; and that the performance of double hashing is quite close to "optimal" (same as separate chaining) unless the table is very full. These and related results can be quite useful in the application of hashing in a practical situation.

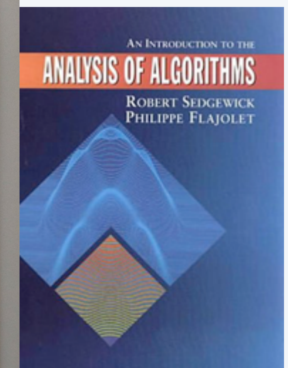
Exercise 8.36 How many keys can be inserted into a linear probing table of size M before the average search cost gets to be greater than $\ln N$?

Exercise 8.37 Compute the exact cost of an unsuccessful search using linear probing for a full table.

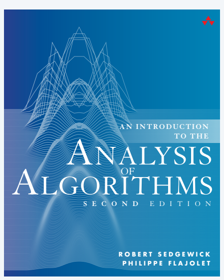
Exercise 8.38 Give an explicit representation for the EGF for the cost of unsuccessful search.

Exercise 8.39 Use the symbolic method to derive the EGF of the number of probes required by linear probing in a successful search, for fixed M.*

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



p. 452



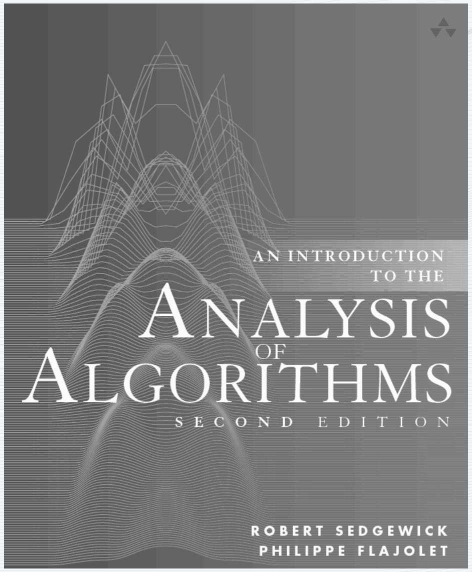
p. 518

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)

ANALYTIC COMBINATORICS

PART ONE



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
...	...	...	...

$M_4 = 64$

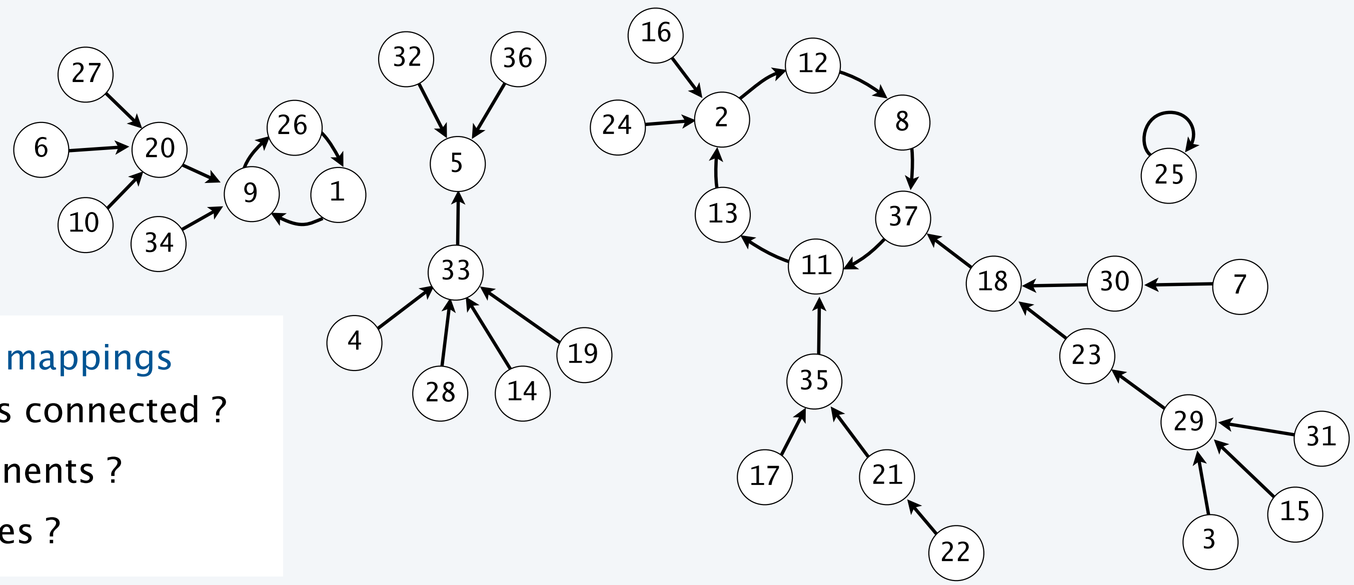
A. N^N

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

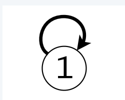


Natural questions about random mappings

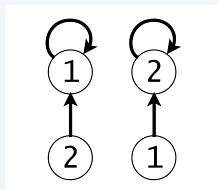
- Probability that the digraph is connected ?
- How many connected components ?
- How many nodes are on cycles ?

Cayley trees

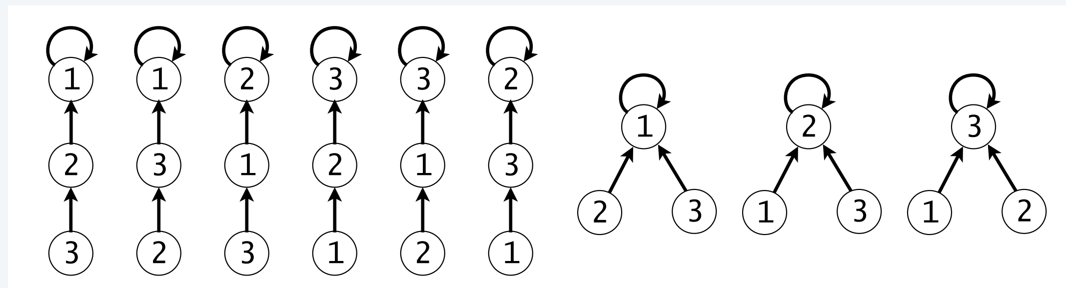
Q. How many different labeled rooted unordered trees of size N?



$C_1 = 1$



$C_2 = 2$

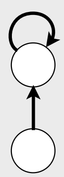


$C_3 = 9$

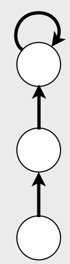
Short form: N-words grouped with unlabeled trees

1

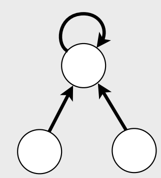
1 1
2 2



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 $H(u)$,

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

$H(u) = u$ 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 $f(u) = u - u^2$

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

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

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 $g_n = \frac{1}{n}[u^{n-1}]\Big(\frac{u}{f(u)}\Big)^n$.

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

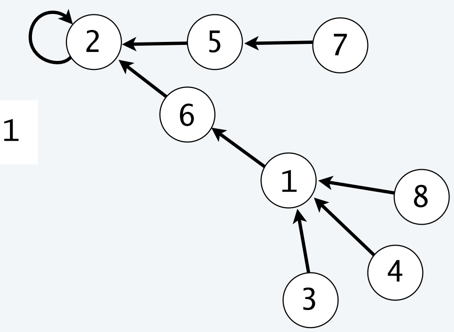
$$\frac{1}{(1-z)^M} = \sum_{k\geq 0}\binom{k+M-1}{M-1}z^k$$

Cayley trees

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

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 $f(u) = u/e^u$

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

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

$$C_N = N![z^N]C(z) = N^{N-1} \checkmark$$

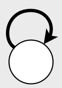
Lagrange Inversion Theorem.

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 $g_n = \frac{1}{n}[u^{n-1}]\Big(\frac{u}{f(u)}\Big)^n$.

Connected components in mappings

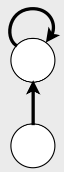
Q. How many different cycles of Cayley trees of size N ?

1

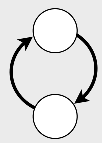


$Y_1 = 1$

1 1
2 2

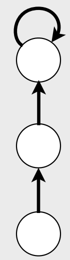


2 1

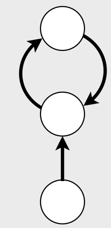


$Y_2 = 3$

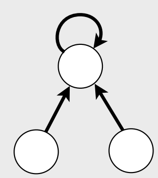
1 1 2
1 3 1
2 2 1
2 3 3
3 1 3
3 2 2



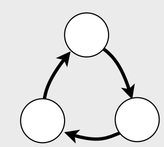
2 1 1
2 1 2
2 3 2
3 1 1
3 3 1
3 3 2



1 1 1
2 2 2
3 3 3



2 3 1
3 1 2



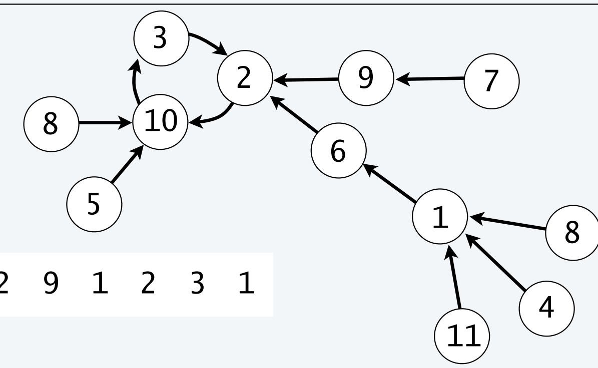
$Y_3 = 17$

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 $\quad Y = CYC(C)$ ⟵ "a component is a cycle of trees"

EGF equation

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

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

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

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

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

Lagrange Inversion Theorem (Bürmann form).

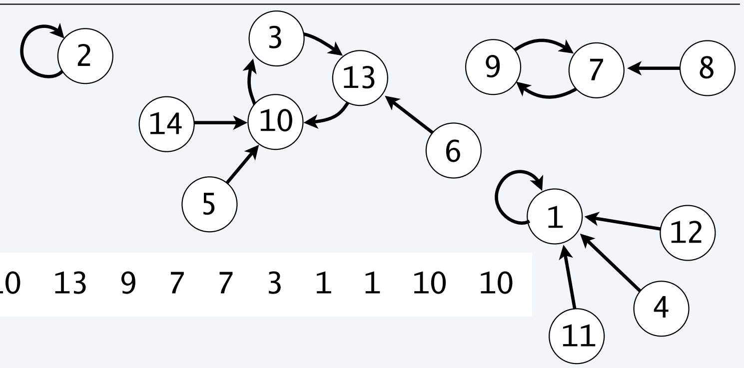
If a GF $g(z) = \sum_{n \geq 1} g_n z^n$ satisfies the equation $z = f(g(z))$ with $f(0) = 0$ and $f'(0) \neq 0$ then, for any function $H(u)$,

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

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 $\quad M = SET(CYC(C))$ ← "a mapping is a set of cycles of trees"

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 $H(u)$,

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

EGF equation

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

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

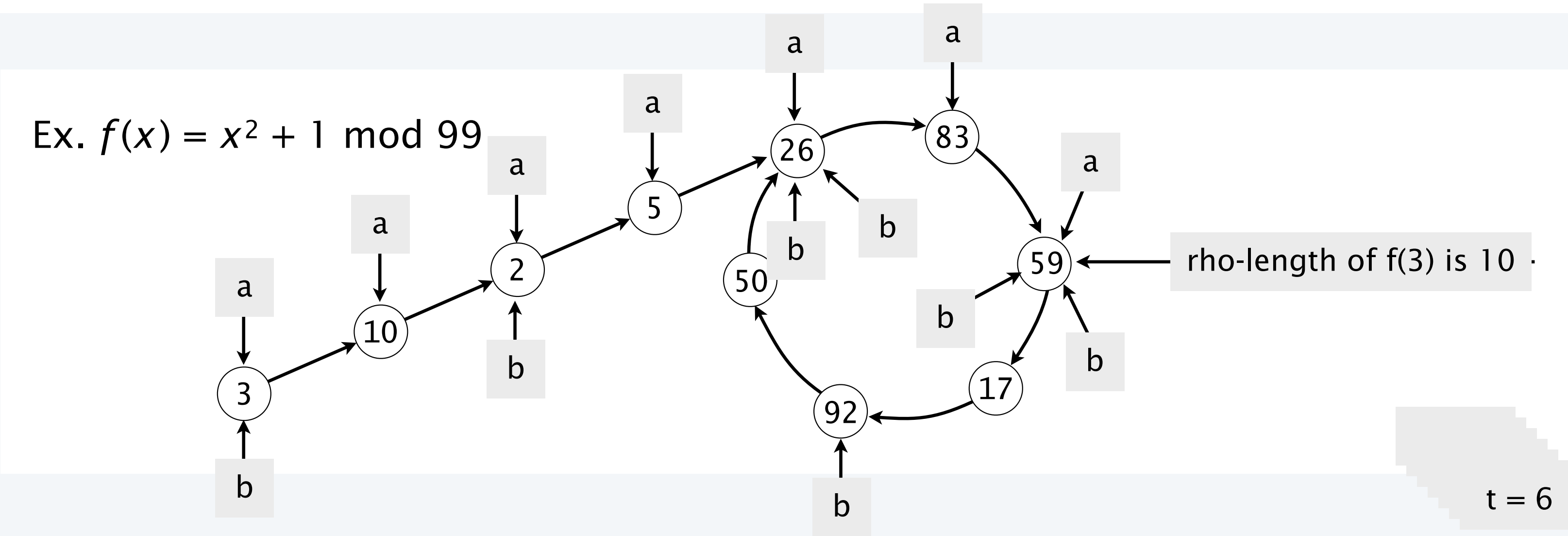
$$[z^N]M(z) = \frac{1}{N}[u^{N-1}]\frac{1}{(1-u)^2}e^{uN}$$

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

$$M_N = N^N \checkmark$$

Rho length

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



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 $M = SET(uCYC(C))$

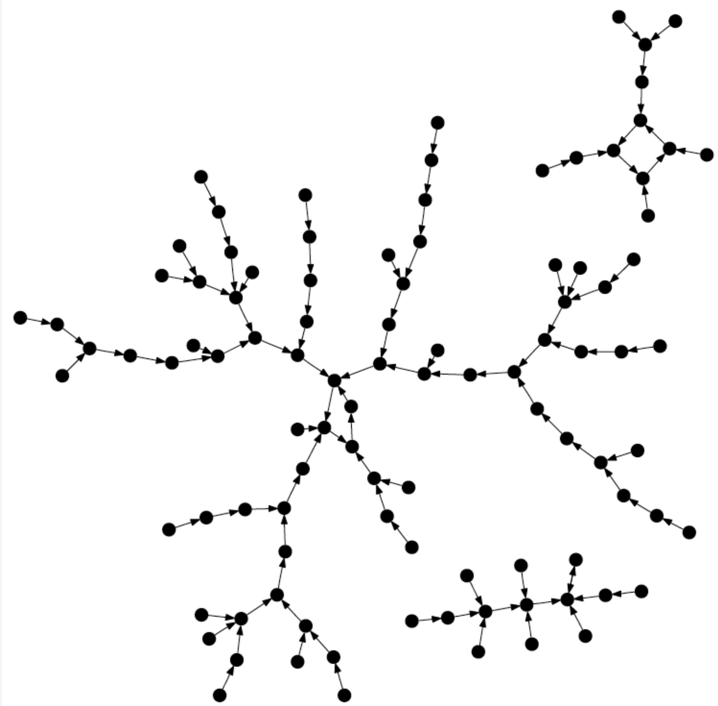
EGF equation $M(z) = \exp\left(u \ln \frac{1}{1 - C(z)}\right) = \frac{1}{(1 - C(z))^u}$

Ex 2. Number of trees (nodes on cycles)

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

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

Stay tuned to Part II for asymptotics.



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. $\sim \sqrt{\pi N/2}$

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 ?

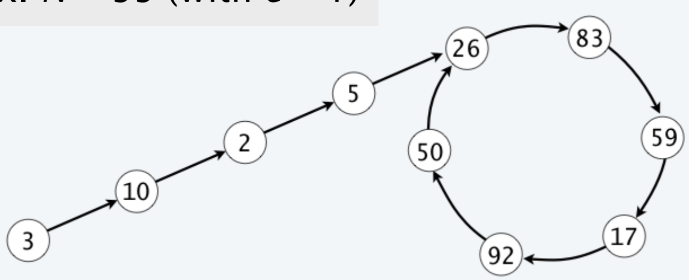
A. Iterate $f(x) = x^2 + c$ until finding a cycle ala Floyd's algorithm. Use a random value of c and start at a random point.

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

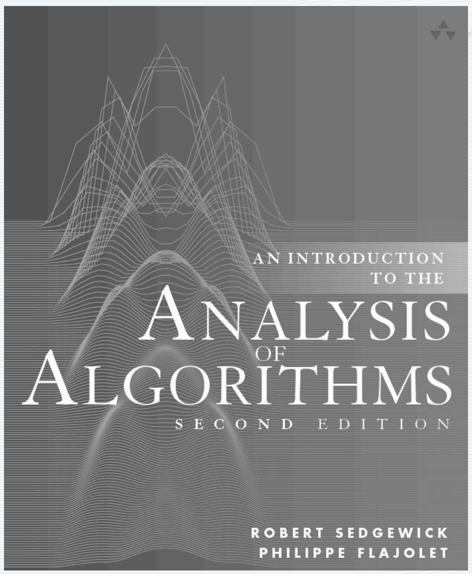
Pollard's algorithm

```
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



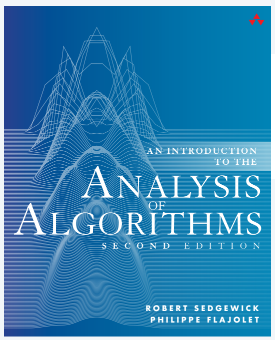
http://aofa.cs.princeton.edu

9. Words and Mappings

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

Exercise 9.5

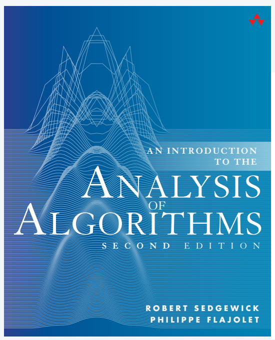
Being *really* sure that the birthday trick will work.



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).

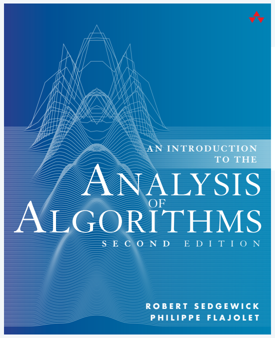


Exercise 9.38 ("Abel's binomial theorem.") Use the result of the previous exercise and the identity $e^{(\alpha+\beta)C(z)} = e^{\alpha C(z)}e^{\beta C(z)}$ to prove that

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

Exercise 9.99

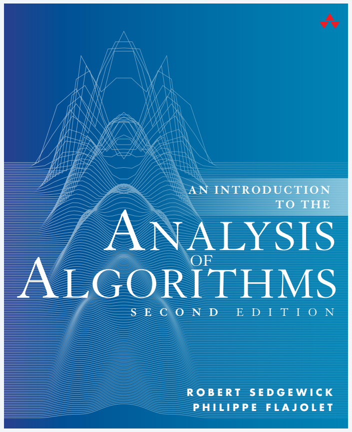
[Not in the book, but should be there.]



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.

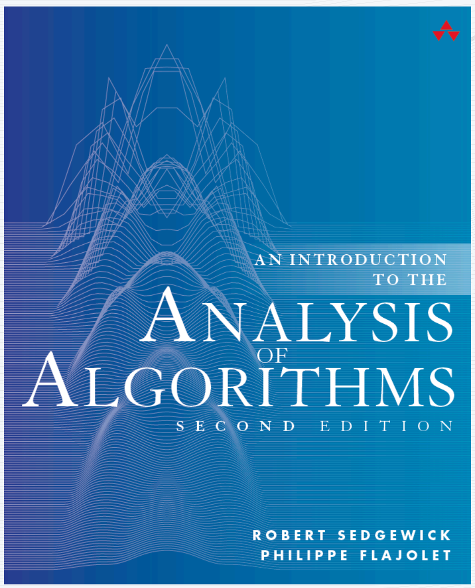
Experiment 9.1. Implement linear probing hashing and run experiments for $N = 1$ million and $M = 900{,}000$ to validate the prediction from Knuth's analysis that about 5.5 probes should be needed, on average, for a successful search.

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.

ANALYTIC COMBINATORICS

PART ONE



http://aofa.cs.princeton.edu

9. Words and Mappings