## CSC 321: Data Structures

## Fall 2016

### Counting and proofs

- mappings, bijection rule
- sequences, product rule, sum rule
- generalized product rule, permutations
- division rule, inclusion/exclusion
- pigeonhole principle
- proof techniques: direct, by-contradiction, by-induction

1

# Bijections

### it is often easier to count one thing by counting another

e.g., to count sheep, count legs and divide by 4

### recall:

a *function* (f : D → R) is a mapping from elements of a domain D to elements of a range R

e.g., sqrt: R → R    abs: R → R+

a *bijective function* or *bijection* is a one-to-one mapping between two sets

e.g., current Creighton students & staff, active netIDs

each student/staff has a unique netID; each netID has a unique student/staff

e.g., f : R → R where f(x) = 2x+1

each x maps to a unique f(x); each f(x) has a unique x

**Bijection Rule:** If there is a bijection f : A → B, then |A| = |B|.

2

## Application: bijection rule

A = # of ways to assign letter grades (A/B/C/D/F) to 10 students
B = # of 14 bit patterns with exactly four 1's

consider grades from A: `2 A's, 4 B's, 1 C, 2 D's, 1 F`

map to a bit pattern from B: `00100001010010`

this is a 1-to-1 mapping, so |A| = |B|

- if we can determine the size of either set, then we know the size of the other
- $|B| = \binom{14}{4}$ = 14!/10!4! = 14*13*12*11/4*3*2*1 = 7*13*11 = 1,001



## Sequences

general strategy:

focus on techniques for counting sequences, then
for each counting problem to be solved, try to map it into sequences

recall:

if $P_1, P_2, \ldots, P_n$ are sets, then $P_1 \times P_2 \times \ldots \times P_n$ is the set of all sequences where the $1^{st}$ term is from $P_1$, $2^{nd}$ term is from $P_2$, …, $n^{th}$ term is from $P_n$

- e.g., C = {red, blue}, N = {1, 2, 3},
  C × N = {red-1, red-2, red-3, blue-1, blue-2, blue-3}

Product Rule: if $P_1, P_2, \ldots, P_n$ are sets, then

$$|P_1 \times P_2 \times \ldots \times P_n| = |P_1| * |P_2| * \ldots * |P_n|$$

# Application: product rule

suppose we are trying to build a computer out of components

P = { i5, i7 }
R = { 2GB, 4GB, 8GB }
C = { 1MB, 2MB, 4MB }

how many combinations of processor, RAM & cache are there to choose from?

- $|P \times R \times C| = |P| * |R| * |C| = 2 * 3 * 3 = 18$

{ i5-2GB-1MB, i5-2GB-2MB, i5-2GB-4MB,
i5-4GB-1MB, i5-4GB-2MB, i5-4GB-4MB,
i5-8GB-1MB, i5-8GB-2MB, i5-8GB-4MB,
i7-2GB-1MB, i7-2GB-2MB, i7-2GB-4MB,
i7-4GB-1MB, i7-4GB-2MB, i7-4GB-4MB,
i7-8GB-1MB, i7-8GB-2MB, i7-8GB-4MB }

5

# Application: product rule

how many possible subsets of a set of N elements?

$S = \{ x_1, x_2, \ldots, x_n \}$

can map each subset into a sequence of N bits: $b_i = 1 \rightarrow x_i$ in subset

$\{ x_1, x_4, x_5 \} \leftrightarrow 10011000\ldots0$

- for an N-element set, count number of N-bit sequences

$| \{0,1\} \times \{0,1\} \times \{0,1\} \times \ldots \times \{0, 1\} | = | \{0,1\} |^n = 2^n$

e.g., S = {a, b, c}
subsets of S = { { }, {a}, {b}, {c}, {a,b}, {a,c}, {b,c}, {a,b,c} }

$|\text{subsets of } S| = 2^{|S|} = 2^3 = 8$

6

## Combining rules

many problems involve a combination of counting methods

Sum Rule: if $S_1, \ldots, S_n$ are disjoint sets, then $|S_1 \cup \ldots \cup S_n| = |S_1|+\ldots+|S_n|$

Division Rule: if $f : A \rightarrow B$ is k-to-1, then $|A| = k * |B|$.

suppose a computer system requires 6-8 character passwords, consisting of letters & digits that must start with a letter

- alpha = { a, b, …, z, A, B, …, Z }
- anum = alpha U { 0, 1, 2, …, 9 }

passwords = (6-char passwords) U (7-char passwords) U (8-char passwords)
= (alpha × $anum^5$) U (alpha × $anum^6$) U (alpha × $anum^7$)

| passwords | = | (alpha × $anum^5$) U (alpha × $anum^6$) U (alpha × $anum^7$) |
= | alpha × $anum^5$ | + | alpha × $anum^6$ | + | alpha × $anum^7$ | [Sum Rule]
= $|alpha|*|anum|^5 + |alpha|*|anum|^6 + |alpha|*|anum|^7$ [Product Rule]
= $52*62^5 + 52*62^6 + 52*62^7$
= 186,125,210,680,448

7

## Generalized product rule

the product rule assumes that the choices are independent

- pick-4 lotto: 4 bins of balls numbered 0-9.
  $| \{0..9\} \times \{0..9\} \times \{0..9\} \times \{0..9\} | = |\{0..9\}|^4 = 10^4 = 1{,}000$

what about a lottery where balls numbered 1-42 are in a bin & draw 4?

Generalized Product Rule: Let S be a set of length-k sequences. If there are:

- $n_1$ possible first entries,
- $n_2$ possible second entries for each first entry,
- $n_3$ possible third entries for each combination of first and second entries, etc.

then:

$| S | = n_1 * n_2 * n_3 * \ldots * n_k$

- | 4-ball draws | = 42 * 41 * 40 * 39 = 2,686,320

8

# Counting subsets

## the lottery problem is an example of a more generic problem

- how many k-element subsets of an n-element set are there?
- e.g., how many 4-ball lottery numbers are there (assuming 42 balls)?
- e.g., how many 5-card poker hands are there (assuming 52 cards)?
- e.g., how many 3-topping pizzas are there (assuming 10 toppings)?

## "n choose k" is such a common expression that it has its own notation: $\binom{n}{k}$

- can map any permutation of n items into a k-element subset by simply taking the first k elements of in the permutation
- this is not a 1-1 mapping though, since any arrangement of the k-element prefix and (n-k)-element suffix yields the same subset
- Division rule → |perms| = |perms of prefix| * |perms of suffix| * $\binom{n}{k}$

$$\rightarrow \quad \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

9

# Pascal's identity

## "n choose k" can also be defined using recursion

- select a "special" element of the list (e.g., lottery ball #1)

|subsets| = |subsets that include special element| + |subsets that don't|

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

e.g., choose 4 out of 42 lottery balls =
choose 3 out of 41 balls (includes 1) +
choose 4 out of 41 balls (doesn't include 1)

```
def choose1(n, k):
    return math.factorial(n)//(math.factorial(k)*math.factorial(n-k))

def choose2(n, k):
    if k == n:
        return 1
    elif k == 1:
        return n
    else:
        return choose2(n-1, k-1) + choose2(n-1, k)
```

10

## Application: hand counting

### how many different 4-of-a-kind hands are there?

- can define a bijection with the sequence:
  rank of matching cards, rank of $5^{th}$ card, suit of $5^{th}$ card
- Generalized Product Rule:

  |sequences| = |rank of matching|*|rank of $5^{th}$|*|suit of $5^{th}$|
  = 13 * 12 * 4
  = 624

- there are "52 choose 5" possible hands = 52!/(47!5!) = 2,598,960
- odds of drawing 4-of-a-kind are 624/2598960 ≈ 1/4165

11

## Application: hand counting

### how many different full house hands are there?

- can define a bijection with the sequence:
  rank of 3, suits of 3, rank of 2, suits of 2
- Generalized Product Rule:

  |sequences| = |rank of 3|*|suits of 3|*|rank of 2|*|suits of 2|
  = 13 * (4 choose 3) * 12 * (4 choose 2)
  = 13 * 4!/(3!1!) * 12 * 4!/(2!2!)
  = 13 * 24/6 * 12 * 24/4
  = 3,744

- odds of drawing a full house are 3744/2598960 ≈ 1/694

12

## Application: hand counting

### how many different 2-pair hands are there?

- can define a mapping with the sequence:
  rank of 1st pair, suits of 1st pair, rank of 2nd pair, suits of 2nd pair,
  rank of extra, suit of extra

- Generalized Product Rule:

  |sequences| = 13 * (4 choose 2) * 12 * (4 choose 2) * 11 * 4
  = 13 * 4!/(2!2!) * 12 * 4!/(2!2!) * 11 * 4
  = 13 * 24/4 * 12 * 24/4 * 11 * 4
  = 247,104

- odds of drawing 2-pairs are 247104/2598960 ≈ 1/11

WRONG: the mapping is not a bijection
3CSQHSAD and QHS3CSAD map to the same hand
2-to-1 mapping, so 247104/2 = 123,552 hands (≈ 1/22)

13

## Set overlap

### recall the Sum Rule: if $S_1$, ..., $S_n$ are disjoint sets, then $|S_1 \cup \ldots \cup S_n| = |S_1|+\ldots+|S_n|$

- but what if the sets are not disjoint?

### suppose JM&C has 30 CSI majors and 20 GDE majors

- that doesn't necessarily mean 50 distinct majors
  total majors = CSI majors + GDE majors – (dual majors)

### Inclusion/Exclusion Rule:

$|S_1 \cup S_2 \cup \ldots \cup S_n|$ = *the sum of the sizes of the individual sets –*
*the sizes of all two-way intersections +*
*the sizes of all three-way intersections –*
*the sizes of all four-way intersections +*
. . .

14

## Pigeonhole principle

suppose your sock drawer contains black, brown, and white socks

- if you grab socks at random, how many must you grab to ensure a match?

Pigeonhole Principle: if |X| > |Y|, then for every total function f : X → Y, there exist two different elements of X that are mapped to the same element of Y

- need to grab 4 socks to make |socks| > |colors|

- how many people must be in a room to ensure at least one shared birthday?
  (only requires 57 for a 99% probability)

15

## Direct proofs

the simplest kind of proof is a logical explanation or demonstration

CLAIM: The best case for sequential search is O(1)

PROOF: Suppose the item to be found is in the first index. Then sequential search will find it on the first check. You can't find something in fewer than one check.

CLAIM: you can add to either end of a doubly-linked list in O(1) time.
PROOF:

- add at front

```
front = new DNode(3, null, front);          → O(1)
if (front.getNext() == null) {              → O(1)
    back = front;                           → O(1)
}
else {
   front.getNext().setPrevious(front);      → O(1)
}
```

- add at back

```
back = new DNode(3, back, null);            → O(1)
if (back.getPrevious() == null) {           → O(1)
    front = back                            → O(1)
}
else {
    back.getPrevious().setNext(back);       → O(1)
}
```

16

## Proof by contradiction

to disprove something, all you need to do is find a counter-example

CLAIM: every set has an even number of elements.

DISPROOF: { 4 }

however, you can't prove a general claim just by showing examples

CLAIM: there is no largest integer

to prove a claim by contradiction

- assume the opposite and find a logical contradiction

CLAIM: there is no largest integer
PROOF: Assume there exists a largest integer. Call that largest integer N.
But N+1 is also an integer (since the sum of two integers is an integer), and N+1 > N.
This contradicts our assumption, so the original claim must be true.

17

## Proof by induction

inductive proofs are closely related to recursion

- prove a parameterized claim by building up from a base case

To prove some property is true for all N ≥ C (for some constant C):

BASE CASE: Show that the property is true for C.
HYPOTHESIS: Assume the property is true for all n < N
INDUCTIVE STEP: Show that that the property is true for N.

CLAIM: 1+2+…+N = N(N+1)/2

BASE CASE: N =1. 1 = 1(1+1)/2 ✓
HYPOTHESIS: Assume the relation holds for all n < N, e.g., 1+2+…+(N-1) = (N-1)N/2.
INDUCTIVE STEP: Then 1+2+…+N = [1+2+…+(N-1)]+N *regrouping*
= (N-1)N/2 + N *by hypothesis*
= ($N^2$ – N)/2 + 2N/2 *simplification*
= ($N^2$ + N)/2 *simplification*
= N(N+1)/2 ✓

18

# Proof by induction

FUNDAMENTAL THEOREM OF ARITHMETIC: every integer $N > 1$ is either prime or the product of primes

BASE CASE: N=2. 2 is prime.

HYPOTHESIS: Assume true for $n < N$.

INDUCTIVE STEP?

Either N is a prime number or not.

If N is prime, the assertion is proven.

If not, then $N = x_1 * x_2 * \ldots * x_k$, where each $x_i < N$ (by the definition of non-prime).

By the induction hypothesis,

each $x_i$ is either prime or a product of primes $x_i = p_{i1} * \ldots * p_{ij}$

Thus, $N = x_1 * x_2 * \ldots * x_k = p_{11} * \ldots * p_{1j} * \ldots * p_{k1} * \ldots * p_{kj}$ is a product of primes.

19