CSCI 2011: Counting and Combinatorics

Chris Kauffman

Last Updated:
Tue Jul 17 16:08:51 CDT 2018

Logistics

Reading: Rosen

- Now: 6.1 - 6.5, 8.5
- Next: 7.1 - 7.4

Assignments

- A06: Post today
- Due Tuesday

Goals

- Induction Wrap-up (Trees)
- Counting/Combinatorics

Counting Stuff

Principles for counting combinations of objects where

Set A has m elements	$\lvert A \rvert = m$
Set B has n elements	$\lvert B \rvert = n$

Product Rule Choosing a pair of items from A, B yields $n \cdot m$ possibilities.

Sum Rule $A \cap B = \emptyset$ (no common elements), choosing a single item from either set A or B yields $n + m$ possibilities.

Difference Rule $A \cap B \neq \emptyset$ (some common elements) choosing a single item from either set A or B yields $n + m - \lvert A \cap B \rvert$ possibilities.

Most folks in college have an intuitive understanding of this already but we'll look at some examples and then exercise it.

Exercise: Counting Telephone #s

- **Local** Telephone Numbers have format `XYZ-ABCD`
- For each digit $X, Y, \ldots, D$ there are 10 choices: `0,1,2,...,9`
- There are 6 digits in a local phone number
- Total local numbers: $\underbrace{10 \cdot 10 \cdot 10}_{\text{7 times}} = 10^7$
- Original North American Telephone Numbering Plan (NATP): digits `X,Y` are restricted to `2,3,...,9`, 8 choices
- Total **valid** local number: $8^2 \cdot 10^5 = 6,400,000$
- Applications of the **Product Rule**

Long Distance Numbers

- Long Distance #s have the form `QRS-XYZ-ABCD`
- Under NATP, `Q` is `2-9`, `R` is `0-1`, `S` is `0-9`
- How many **valid** long distance #s are there?

Answers: Counting Telephone #s

Long Distance Numbers

- Long Distance #s have the form `QRS-XYZ-ABCD`
- Under NATP, `Q` is `2-9`, `R` is `0-1`, `S` is `0-9`
- How many **valid** long distance #s are there?
  - \$8$\cdot\ 2 \cdot 10 \cdot (8^2 \cdot 10^5) = 1{,}024{,}000{,}000$

Exercise: Counting Functions

- A function from m elements to n elements maps each of the m to one of the n
- So m choices with n possibilities each
- Total possible functions: $\underbrace{n \cdot n \cdot \ldots \cdot n}_{m \text{ times}} = n^m$
- If the function is **one-to-one**, $m \leq n$ and once an element is chosen, can't pick it again
- Leads to $\underbrace{(n) \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-m+1)}_{m \text{ times}}$
- Applications of the **Product Rule**

One-to-One Calculations

- How many one-to-one functions are there from
  - 3 elements to 7 elements
  - 8 elements to 12 elements
- Derive an expression for this using the factorial notation

Answers: Counting Functions

One-to-One Calculations

- How many one-to-one functions are there from
  - 3 elements to 7 elements: $7 \cdot 6 \cdot 5 = 210$
  - 8 elements to 12 elements: $12 \cdot 11 \cdot 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 = 19,958,400$
- Derive an expression for this using the factorial notation

$$\frac{n!}{(n-m)!} = \frac{(n) \cdot (n-1) \cdot \ldots \cdot (n-m+1) \cdot (n-m) \cdot (n-m-1) \cdot \ldots \cdot 2 \cdot 1}{(n-m) \cdot (n-m-1) \cdot \ldots \cdot 2 \cdot 1}$$

$$= \underbrace{(n) \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-m+1)}_{n-m \text{ times}}$$

Exercise: Project Choices

- CS course requires a single final project from one of three categories with differing lists projects

Subject	Count
AI	23 projects
Robotics	15 projects
Vision	19 projects

- No project appears on two lists
- Total possibilities: $23 + 15 + 19 = 57$
- Application of the **Sum Rule**

- For Extra Credit, student can do 2 projects from different lists
- Possibilities
  - AI/Robotics: $23 \cdot 15 = 345$
  - AI/Vision: $23 \cdot 19 = 437$
  - Robotics/Vision: $23 \cdot 19 = 285$
  - Total: $345 + 437 + 285 = 1067$

More Choices

How many possibilities if 2 different projects

- Both could be from the same list?
- Both **must** be from the same list?

Important: Picking projects (A,B) same as (B,A)

Answers: Project Choices

How many possibilities if 2 different projects

Subject	Count
AI	23 projects
Robotics	15 projects
Vision	19 projects

Both could be from the same list?

- 57 total projects
- Two choices
  1. 57 possibilities
  2. 56 possibilities

  Two orderings of projects
- $57 \cdot 56/2 = 3192/2 = 1596$

Both **must** be from the same list?

Pick category, then pick two projects

AI	$24 \cdot 23/2 = 276$
Robotics	$15 \cdot 14/2 = 105$
Vision	$19 \cdot 18/2 = 171$
Total	(210+342+552) / 2 $= 552$

Exercise: Difference Rule Applications

- Startup posts 2 jobs: Web Developer and Database Administrator
- Receives applicants for both jobs with some redundancy

Web Dev	220 Applicants
DB Admin	147 Applicants
Both	57 Applicants

- The Total number of applicants to be evaluated:

$$220 + 147 - 57 = 310$$

Subject	Count
AI	23 projects
Robotics	15 projects
Vision	19 projects
AI/Robotics	3 in common
AI/Vision	8 in common
Robotics/Vision	4 in common
On all	2 in common

How many possibilities if:

- Pick 2 unique projects, 2nd not on first list
- Pick 1 project from any list
  Hint: Careful with subtracting twice

Answers: Difference Rule Applications

Subject	Count
AI	23 projects
Robotics	15 projects
Vision	19 projects
AI/Robotics	3 in common
AI/Vision	8 in common
Robotics/Vision	4 in common
On all	2 in common

How many possibilities if:

- Pick 2 unique projects, 2nd not on first list

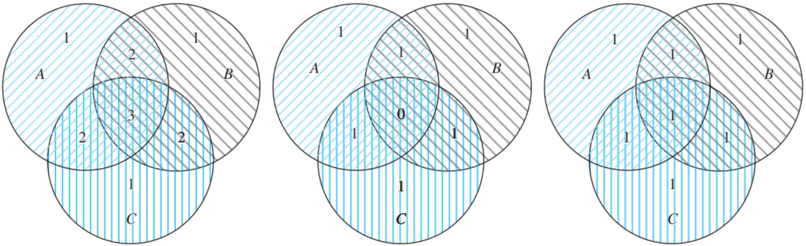
AI/Robotics	$23 \cdot (15 - 3)$	$= 276$
AI/Vision	$23 \cdot (19 - 8)$	$= 253$
Robotics/Vision	$15 \cdot (19 - 4)$	$= 225$
Total	$276{+}253{+}225$	$= 754$

- Pick 1 project from any list
  Total possible: $(23 + 15 + 19) - (3 + 8 + 4) + \mathbf{2} = 46$
  Last term re-adds overlap of all, see next slide

Principle of Inclusion/Exclusion (Rosen Sec 8.5)

- Following identity holds for sets A, B as part of the Difference Rule: $|A \cup B| = |A| + |B| - |A \cap B|$
- This Case Involves 2 sets
  - Intersection added by both $|A|$ and $|B|$
  - Subtracted to correct for this
- More sets requires more careful consideration of which overlaps are added multiple times
- Example: Size of Union of 3 sets gives $|A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$

3-way intersection calculation



(a) Count of elements by $|A|+|B|+|C|$

(b) Count of elements by $|A|+|B|+|C|-|A\cap B|-|A\cap C|-|B\cap C|$

(c) Count of elements by $|A|+|B|+|C|-|A\cap B|-|A\cap C|-|B\cap C|+|A\cap B\cap C|$

FIGURE 3 Finding a Formula for the Number of Elements in the Union of Three Sets.

$$|A\cup B\cup C| = |A|+|B|+|C|-|A\cap B|-|A\cap C|-|B\cap C|+|A\cap B\cap C|$$

Examples of Inclusion/Exclusion

- This is the simplest case: 2 sets with intersection size known
  - How many ints 1 to 1000 are divisible by 7 or 11?
  - Apply $|A \cup B| = |A| + |B| - |A \cap B|$
  - $\lfloor 1000/7 \rfloor + \lfloor 1000/11 \rfloor - \lfloor 1000/(7 \cdot 11) \rfloor$
- More complex cases require care:
  - How many ints 1 to 1000 are divisible by 7 or 11 or 13? Apply...
  - $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$

$$\begin{aligned}
&\lfloor 1000/7 \rfloor + \lfloor 1000/11 \rfloor + \lfloor 1000/13 \rfloor \\
&- (\lfloor 1000/(7 \cdot 11) \rfloor + \lfloor 1000/(7 \cdot 13) \rfloor + \lfloor 1000/(11 \cdot 13) \rfloor) \\
&+ \lfloor 1000/(7 \cdot 11 \cdot 13) \rfloor
\end{aligned}$$

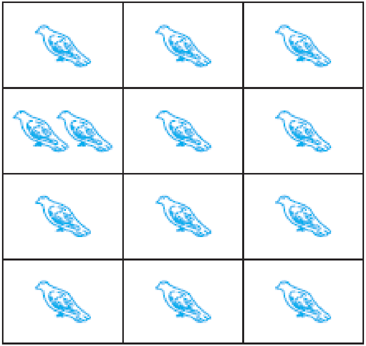
- Predict: Divisible by 7,11,13,or 17? (intersection of 4 sets)

Pigeonholing

The Pigeonhole Principle

Standard: If k is a positive integer and $k+1$ or more objects are placed into k boxes, there is at least one box containing two or more of the objects.

Generalized: If N objects are placed into k boxes, there is at least one box containing $\lceil N/k \rceil$ objects.



Examples of Pigeonholing

Example: Labspace Sharing

A computer lab classroom has 15 desktop computers and 17 students.

- If the class is fully attended, then there is at least 1 desktop with 2 students sharing it.

But when is lab ever fully attended?

Example: Birth Months

In a group of 100 people, what is the **minimum** number that share a birth month?

- By the pigeonhole principle, 100 people and 12 boxes (months), so at least $\lceil 100/12 \rceil = 9$ must have the same birth month.

Frenemies: An Intricate Example of Pigeonholing

In group of 6 people called A, B, C, D, E, F, each pair of people is either friends or an enemies. **Prove** that the group must have at least

3 mutual friends OR 3 mutual enemies

in the group. For example, all 6 could be friends or all 6 could be enemies. **Proof:**

1. Consider A's relationship to the remaining 5 people each of which are in one of 2 groups: friend of A or enemy of A.
2. By the pigeonhole principle, one of these two groups must have $\lceil 5/2 \rceil = 3$ people in it.
3. **Without loss of generality**, assume that B, C, D are friends of A while E, F are enemies.
4. If B, C are friends, then A, B, C are 3 mutual friends
5. If B, D are friends, then A, B, D are 3 mutual friends
6. If C, D are friends, then A, C, D are 3 mutual friends
7. If none of (4-6) are true, then B, C, D are 3 mutual enemies
8. By a combination of (4-7), have shown that there must be a 3 mutual friends or 3 mutual enemies. ■

Combinatorics: Combinations and Permutations

Many problems involve selection and ordering of objects

I have 7 shirts and 4 pairs of pants. How many schedules of outfits can I plan that do not repeat the same outfit in the next 5 days?

Common enough to have some associated terminology and techniques

$P(n, r)$ Permutations Number of **ORDERED** selection of r objects from a collection of n objects without repetition

$C(n, r)$ Combinations Number of **UNORDERED** selection of r objects from a collection of n objects without repetition

Permutations

$$\begin{aligned} P(n,r) &= \frac{n!}{(n-r)!} \\ &= \frac{(n)\cdot(n-1)\cdot\ldots\cdot 2\cdot 1}{(n-r)\cdot(n-r-1)\cdot\ldots\cdot 2\cdot 1} \\ &= \underbrace{(n)\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot(n-r+1)}_{n-r \text{ times}} \end{aligned}$$

- This should look familiar based on our earlier discussion
- Orderings of r objects selected without repetition from n

Example

I have 8 shirts and 5 work days. How many different orderings of shirts can I wear without repeating on the 5 days.

- 5 days to choose, 8 choices **initially**
- 4 days left and 7 choices, 3 days left and 6 choices...
- $8\cdot 7\cdot 6\cdot 5\cdot 4 = P(8,5) = (8!)/(3!) = 6720$ orderings

Combinations

$$C(n, r) = \frac{P(n, r)}{r!} = \frac{n!}{r!(n-r)!}$$

- Number of subsets of size r selectable from set of size n
- Order doesn't matter: Permutations discounting orderings

Example

I have 8 shirts and 5 vacation days. How many different combinations of shirts can end up in my suitcase.

- 5 days to choose, 8 choices **initially**
- 4 days left and 7 choices, 3 days left and 6 choices...
- $8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 = P(8, 5) = (8!)/(3!) = 6720$ orderings
- $5! = 120$ orderings of 5 shirts
- $C(8, 5) = P(8, 5)/5! = 56$ combinations

Exercise: Permutations or Combinations?

Give answers in permutation/combination form.

1. How many length 16 bit strings have exactly 12 ones?
2. How many length 5 strings can be formed from letters `ABCDEFGH`?
3. How many length 16 bit strings have an odd number of 1's?
4. How many length 5 strings that contain characters `ABC` in any order can be formed from letters `ABCDEFGH`? *Hint: two groups ABC, and DEFGH*
5. How many length 16 bit strings are there?

Answers: Permutations or Combinations?

1. How many length 16 bit strings have exactly 12 ones?
   - Pick 12 indices at which to put 1's, order of index selection doesn't matter
   - $C(16, 12) = 1820$
2. How many length 5 strings can be formed from letters `ABCDEFGH`?
   - Order Matters
   - $P(8, 5) = 6720$
3. How many length 16 bit strings have an odd number of 1's?
   - $C(16, 1) + C(16, 3) + C(16, 5) + \cdots + C(16, 5)$
4. How many length 5 strings that contain characters `ABC` in any order can be formed from letters `ABCDEFGH`? *Hint: two groups ABC, and DEFGH*
   - *Sol1*: Pick 2 from group 2 DEFGH, then order 5 letters: $C(5, 2) \cdot P(5, 5)$
   - *Sol2*: Pick 3 indices from 5 for ABC, order ABC, pick remaining 2 chars from 5: $C(5, 3) \cdot P(3, 3) \cdot P(5, 2)$
5. How many length 16 bit strings are there?
   - 0 or 1 for each digit: $\underbrace{2 \cdot 2 \cdots 2}_{\text{16 times}} = 2^{16}$

An Important Identity on Combinations

$$C(n, r) = C(n, n - r)$$

- Symmetry in combination selection
- Proof by definition using factorial

$$C(n, r) = \frac{n!}{r!(n-r)!} \quad = \frac{n!}{(n-r)!(n-(n-r))!} = C(n, n-r)$$

- Intuitive idea choose what to take OR what to leave
  - 7 outfits, pack for a 5 day vacation
  - Choose 5 outfits to TAKE with: $C(7, 5) = 21$
    OR
  - Choose 2 outfits to LEAVE home $C(7, 2) = 21$
  - Same number of possibilities either way

Binomial Coefficients and the Binomial Theorem

- Notation convention:

$$C(n, r) \equiv \binom{n}{r}$$

- $\binom{n}{r}$ called the **Binomial Coefficients** because they show up as coefficients in binomials for different powers of leading variable.

$$\begin{aligned}(x+y)^4 &= (x+y)(x+y)(x+y)(x+y) \\ &= 1x^4 + 4x^3y^1 + 6x^2y^2 + 4x^1y^3 + 1y^4 \\ &= \binom{4}{0}x^4 + \binom{4}{1}x^3y^1 + \binom{4}{2}x^2y^2 + \binom{4}{1}x^1y^3 + \binom{4}{0}y^4\end{aligned}$$

- In general, **Binomial Theorem** States

$$(x+y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j} y^j$$

- Sometimes useful in proving counting identities

Pascal's Identity and Triangle

Pascal's Identity:

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

- Combinatorial Argument or Algebraic Manipulation can prove this
- Basis for a recursive definition of Binomial coefficients AND
- Basis for Pascal's Triangle, a visualization of Binomial Coefficients in a satisfying geometric pattern

Next slide

- Note the nice recursive structure to Pascal's Triangle
- Base cases of $\binom{n}{0} = 1$ and $\binom{n}{n} = 1$
- Tip of triangle is $\binom{0}{0} = 1$, a base case
- Each row element is defined by adding two elements from the previous row

Pascal's Triangle: (a) Combinations Form (b) Numeric

$$
\begin{array}{c}
\binom{0}{0} \\
\binom{1}{0} \quad \binom{1}{1} \\
\binom{2}{0} \quad \binom{2}{1} \quad \binom{2}{2} \\
\binom{3}{0} \quad \binom{3}{1} \quad \binom{3}{2} \quad \binom{3}{3} \\
\binom{4}{0} \quad \binom{4}{1} \quad \binom{4}{2} \quad \binom{4}{3} \quad \binom{4}{4} \\
\binom{5}{0} \quad \binom{5}{1} \quad \binom{5}{2} \quad \binom{5}{3} \quad \binom{5}{4} \quad \binom{5}{5} \\
\binom{6}{0} \quad \binom{6}{1} \quad \binom{6}{2} \quad \binom{6}{3} \quad \binom{6}{4} \quad \binom{6}{5} \quad \binom{6}{6} \\
\binom{7}{0} \quad \binom{7}{1} \quad \binom{7}{2} \quad \binom{7}{3} \quad \binom{7}{4} \quad \binom{7}{5} \quad \binom{7}{6} \quad \binom{7}{7} \\
\binom{8}{0} \quad \binom{8}{1} \quad \binom{8}{2} \quad \binom{8}{3} \quad \binom{8}{4} \quad \binom{8}{5} \quad \binom{8}{6} \quad \binom{8}{7} \quad \binom{8}{8} \\
\cdots
\end{array}
$$

(a)

By Pascal's identity:

$$\binom{6}{4} + \binom{6}{5} = \binom{7}{5}$$

$$
\begin{array}{c}
1 \\
1 \quad 1 \\
1 \quad 2 \quad 1 \\
1 \quad 3 \quad 3 \quad 1 \\
1 \quad 4 \quad 6 \quad 4 \quad 1 \\
1 \quad 5 \quad 10 \quad 10 \quad 5 \quad 1 \\
1 \quad 6 \quad 15 \quad 20 \quad 15 \quad 6 \quad 1 \\
1 \quad 7 \quad 21 \quad 35 \quad 35 \quad 21 \quad 7 \quad 1 \\
1 \quad 8 \quad 28 \quad 56 \quad 70 \quad 56 \quad 28 \quad 8 \quad 1 \\
\cdots
\end{array}
$$

(b)

Permutations/Combinations with Repetition

- Combinations/Permutations $C(n, r)/P(n, r)$ account for **no repetition**: can't select an outfit twice
- Reality has many cases where this is not so

Example Cookie Shop

- Cookie shop has $n = 4$ kinds of cookies, ABCD
- Customer wants $r = 6$ cookies total
- Can duplicate cookies like AABBCD
- How many possibilities are there?
  - If order DOES matters?
  - Order DOESN'T matter: AABBCD $\equiv$ ABBCDA?

Formulae for Perm/Comb with Repetition

Ordered Repetition

- Permutations, n^r possibilities
- $n = 4$ kinds of cookies, $r = 6$ choices
- $4^6 = 4096$ possible orders

TABLE 1 **Combinations and Permutations With and Without Repetition.**

Type	*Repetition Allowed?*	*Formula*
r-permutations	No	$\frac{n!}{(n-r)!}$
r-combinations	No	$\frac{n!}{r!\,(n-r)!}$
r-permutations	Yes	n^r
r-combinations	Yes	$\frac{(n+r-1)!}{r!\,(n-1)!}$

Unordered Repetition

- Combinations with repetition, apply the following formula

$$C(n + r - 1, r)$$

  for n items and r choices with repetition
- With $n = 4$ and $r = 6$ have

$$C(4 + 6 - 1, 4) = 84$$

Permutation/Combination Code

- Quasi-common interview questions
  1. `print_perm(N)`: Write a procedure that prints all permutations of 1 to N.
  2. `print_comb(N)`: Write a procedure that prints all subsets of the numbers 1 to N.
- Also useful for some search procedures.
- Surprisingly Tricky so some practice is good for you
- Several possibilities for these exist and are worth considering.
  - Recursive implementations
  - Iterative only-implementations

Exercise: Recursive Permutations

```
print_perm(N){
  create array cur[] = {}  # empty
  create array rem[] = {1,2,3,...N}
  print_perm_helper(cur,rem)
}

print_perm_helper(int cur[],
                  int rem[])
{
  if(rem is empty){
    print cur;
    return;
  }
  for(int i=0; i<length(rem); i++){
    int c = rem[i]
    append c to end of cur[]
    remove rem[i] shifting over
    print_perm_helper(cur, rem)
    remove cur[length(cur)-1]   # c removed
    insert c at rem[i] shifting over
  }
  return
}
```

- Note the use of recursive helper
- Demonstrate how this code works when `print_perm(3)` is called
- Speculate on Computational Complexity of the code

Answers: Recursive Permutations Execution

```
print_perm(N){
  create array cur[] = {}  # empty
  create array rem[] = {1,2,3,...N}
  print_perm_helper(cur,rem)
}

print_perm_helper(int cur[],
                  int rem[])
{
  if(rem is empty){
    print cur;
    return;
  }
  for(int i=0; i<length(rem); i++){
    int c = rem[i]
    append c to end of cur[]
    remove rem[i] shifting over
    print_perm_helper(cur, rem)
    remove cur[length(cur)-1]   # c removed
    insert c at rem[i] shifting over
  }
  return
}
```

```
print_perm(3)
helper()
  cur = {}
  rem = {1,2,3}
  i = 0
    cur = {1}
    rem = {2,3}
    helper()
      i = 0
        cur = {1,2}
        rem = {3}
        helper()
          i = 0
            cur = {1,2,3}
            rem = {}
            helper()
              print {1,2,3}  ---
      i = 1
        cur = {1,3}
        rem = {2}
        helper()
           i = 0
            cur = {1,3,2}
            rem = {}
            helper()
              print {1,3,2} ---
   i = 1
     cur = {2}
     rem = {1,3}
     helper()
       i = 0
         cur = {2,1}
         rem = {3}
         helper()
```

Answers: Recursive Permutations Complexity

```
1  print_perm(N){
2    create array cur[] = {}  # empty
3    create array rem[] = {1,2,3,...N}
4    print_perm_helper(cur,rem)
5  }
6
7  print_perm_helper(int cur[],
8                    int rem[])
9  {
10   if(rem is empty){
11     print cur;
12     return;
13   }
14   for(int i=0; i<length(rem); i++){
15     int c = rem[i]
16     append c to end of cur[]
17     remove rem[i] shifting over
18     print_perm_helper(cur, rem)
19     remove cur[length(cur)-1]   # c removed
20     insert c at rem[i] shifting over
21   }
22   return
23 }
```

- `print_perm(N)` setup is $O(N)$ to create arrays
- Base case is at worst $O(N)$ to print array
- Loop from 14-20 is $O(N)$ iterations
- Append is $O(1)$
- Insert/Remove with shifting: $O(N)$
- Loop is at least $O(N^2)$ BUT...
- Analyzing recursion is difficult without recurrence relations / Master Theorem

Iterative Versions

- Part of the complexity coming from `print_per_helper()` comes from moving repeatedly shifting array contents
- Possible to avoid this with swapping and greate care though the code becomes much harder to reason about
- Book provides a "next permutation" function which re-orders an array of 1-N to the next permutation
- What is it's complexity?
- How many permutations are there for N objects?
- Reason about complexity of printing all permutations?

Exercise: Next Permutation Procedure

procedure *next permutation*($a_1 a_2 \ldots a_n$: permutation of
$\{1, 2, \ldots, n\}$ not equal to $n \;\; n-1 \;\; \ldots \;\; 2 \;\; 1$)
$j := n - 1$
while $a_j > a_{j+1}$
  $j := j - 1$
$\{j$ is the largest subscript with $a_j < a_{j+1}\}$
$k := n$
while $a_j > a_k$
  $k := k - 1$
$\{a_k$ is the smallest integer greater than a_j to the right of $a_j\}$
interchange a_j and a_k
$r := n$
$s := j + 1$
while $r > s$
  interchange a_r and a_s
  $r := r - 1$
  $s := s + 1$
{this puts the tail end of the permutation after the jth position in increasing order}
$\{a_1 a_2 \ldots a_n$ is now the next permutation$\}$

Answers: Next Permutation Procedure

- What is the complexity `next_permutation()`?
  - $O(N)$: all loops iterate through at most N elements of the permtuation
- How many permutations are there ?
  - $N!$ permutations of N objects
- Reason about complexity of printing all permutations?
  - $O(N \cdot N!)$ total complexity to print all permutations

Combinations Code

- Combinations code is easier
- Realize: combination is a subset of `{1,2,...,N}`
- Each element is either in or out via 1 or 0
- Use a **bitstring** of length N for this and iterate over all possible bit strings
- Next combination is found by adding 1 to bitstring
- For short bitstrings (32,64,128), this is integer addition
- For arbitrary length N bitstring, $O(N)$ operation

```
print_combs(N){
  cur_bs = bitstring of N 0's
  while(cur_bs < pow(2,N)){
    for(i=0; i<N; i++){
      if(cur_bs[i] == 1){
        print i+1
      }
    }                   # O(N) iters
    add 1 to cur_bs     # O(N)
  }                     # O(2^N) iters
}                       # O(N * 2^N)
```