COMP170
Discrete MathematicalTools
for Computer Science

COMP170
Discrete MathematicalTools for Computer Science
Lecture 1

Version 2. Last updated, Sep 01, 2005

Discrete Math for Computer Science
K. Bogart, C. Stein and R.L. Drysdale
Section 1.1, pp. 1-8

Counting

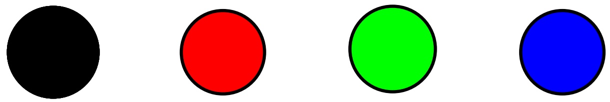
Counting

What's the big deal?   Counting is easy, isn't it?

Counting

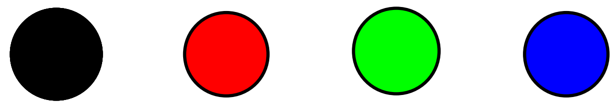
What's the big deal?   Counting is easy, isn't it?

How many different ways are there to choose 2 balls from



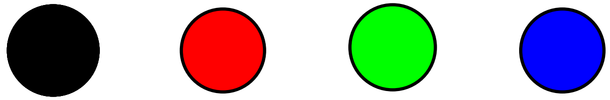
Counting

What's the big deal? Counting is easy, isn't it?

How many different ways are there to choose 2 balls from  $\boxed{6}$

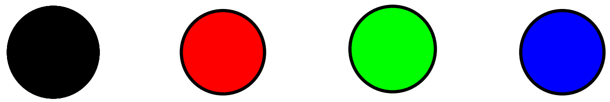
Counting

What's the big deal? Counting is easy, isn't it?

How many different ways are there to choose 2 balls from  $\boxed{6}$

Counting

What's the big deal? Counting is easy, isn't it?

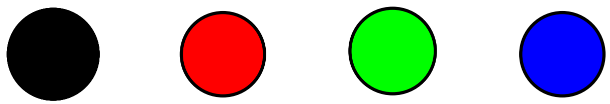
How many different ways are there to choose 2 balls from  $\boxed{6}$

How many different ways are there to choose 2 students from a class of 4 students?

Counting

What's the big deal?   Counting is easy, isn't it?

How many different ways are there to choose 2 balls from



$\boxed{6}$

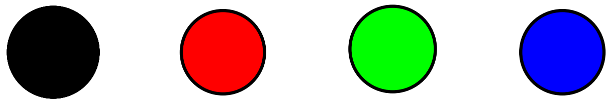
How many different ways are there to choose 2 students from a class of 4 students?

$\boxed{6}$

Same as balls

Counting

What's the big deal? Counting is easy, isn't it?

How many different ways are there to choose 2 balls from  $\boxed{6}$

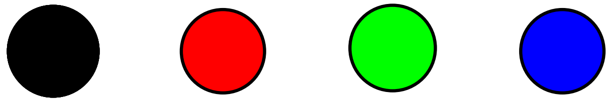
How many different ways are there to choose 2 students from a class of 5 students?

Might still be able to list all

Counting

What's the big deal? Counting is easy, isn't it?

How many different ways are there to choose 2 balls from



$\boxed{6}$

How many different ways are there to choose 2 students from a class of 5 students?

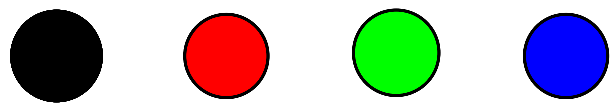
$\boxed{10}$

Might still be able to list all

Counting

What's the big deal? Counting is easy, isn't it?

How many different ways are there to choose 2 balls from



$\boxed{6}$

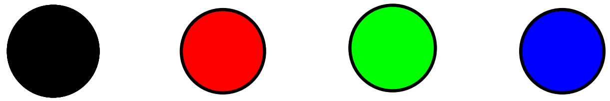
How many different ways are there to choose 2 students from a class of 100 students?

Too many to list

Counting

What's the big deal? Counting is easy, isn't it?

How many different ways are there to choose 2 balls from



$\boxed{6}$

How many different ways are there to choose 2 students from a class of 100 students?

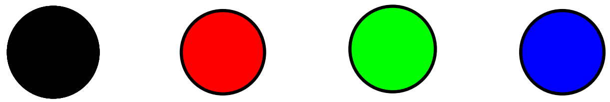
$\boxed{4950}$

Too many to list

Counting

What's the big deal?   Counting is easy, isn't it?

How many different ways are there to choose 2 balls from



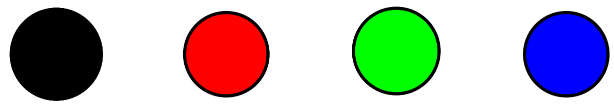
$\boxed{6}$

How many different ways are there to choose 6 numbers out of $1 \dots 49$?

Hong Kong Mark 6!

Counting

What's the big deal? Counting is easy, isn't it?

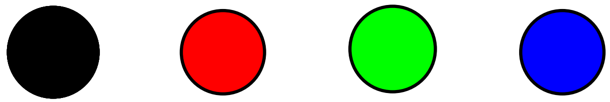
How many different ways are there to choose 2 balls from  $\boxed{6}$

How many different ways are there to choose 6 numbers out of $1 \ldots 49$? $\boxed{13{,}983{,}816}$

Hong Kong Mark 6!

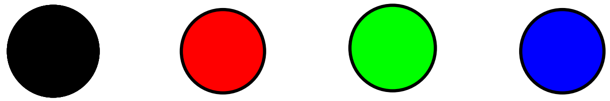
Counting

What's the big deal? Counting is easy, isn't it?

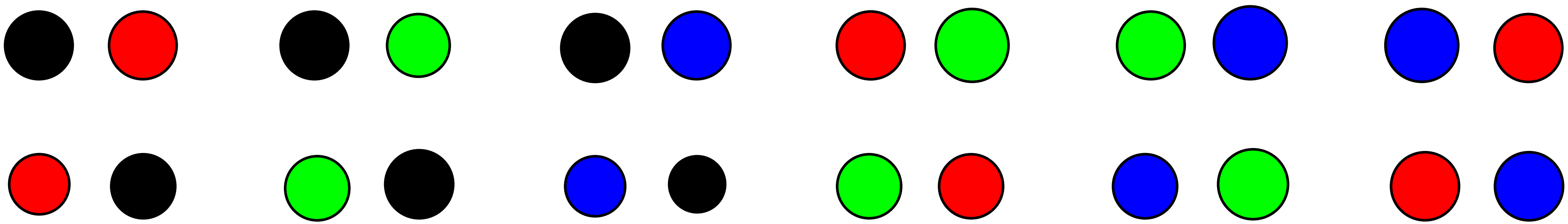
How many different ways are
there to choose 2 balls from 
when order counts

Counting

What's the big deal? Counting is easy, isn't it?

How many different ways are there to choose 2 balls from  $\boxed{12}$

when order counts



In Computer Science we often need to count objects.

Sometimes it's the number of steps a computer program takes

This lets us compare runtimes of different programs.

Sometimes, it's the number of objects of a particular type, e.g., passwords containing between 6-10 characters

This lets us evaluate security. The more passwords available, the lower the chance that someone can guess a password

1.1 Basic Counting

1.1 Basic Counting

- The Sum Principle and set notation

1.1 Basic Counting

- The Sum Principle and set notation
- Abstraction

1.1 Basic Counting

- The Sum Principle and set notation
- Abstraction
- Summing Consecutive Integers

1.1 Basic Counting

- The Sum Principle and set notation
- Abstraction
- Summing Consecutive Integers
- The Product Principle

1.1 Basic Counting

- The Sum Principle and set notation
- Abstraction
- Summing Consecutive Integers
- The Product Principle
- Two-Element Subsets

The Sum Principle

Start with an exercise illustrating the sum principle.

Consider the following loop from **selection-sort**, (comp171), which sorts a list of items

```
(1) for i = 1 to n-1
(2)    for j = i+1 to n
(3)       if (A[i] > A[j])
(4)          exchange A[i] and A[j]
```

The Sum Principle

Start with an exercise illustrating the sum principle.

Consider the following loop from **selection-sort**, (comp171), which sorts a list of items

```
(1) for i = 1 to n-1
(2)   for j = i+1 to n
(3)     if (A[i] > A[j])
(4)       exchange A[i] and A[j]
```

If you've never programmed before **Don't worry!**

This is *Pseudocode*; You will learn more in the tutorial

The Sum Principle

Start with an exercise illustrating the sum principle.

Consider the following loop from **selection-sort**, (comp171), which sorts a list of items

```
(1) for i = 1 to n-1
(2)   for j = i+1 to n
(3)     if (A[i] > A[j])
(4)       exchange A[i] and A[j]
```

How many times is the comparison

A[i] $>$ **A[j]**

made in line 3?

```
for i = 1 to n-1
    for j = i+1 to n
        if (A[i] > A[j])
            exchange A[i] and A[j]
```

```
(1) for i = 1 to n-1
(2)    for j = i+1 to n
(3)       if (A[i] > A[j])
(4)          exchange A[i] and A[j]
```

Lines 2–4 are executed $n - 1$ times, once for each i between 1 and $n - 1$.

```
(1) for i = 1 to n-1
(2)    for j = i+1 to n
(3)       if (A[i] > A[j])
(4)          exchange A[i] and A[j]
```

Lines 2–4 are executed $n-1$ times,
once for each i between 1 and $n-1$.

First time,	$n-1$ comparisons.
Second time,	$n-2$ comparisons.
ith time,	$n-i$ comparisons.
$(n-1)$st time,	1 comparison.

Thus, total number of comparisons is

$$(n-1)+(n-2)+\cdots+1.$$

What did we really do?

What did we really do?

```
(1) for i = 1 to n-1
(2)   for j = i+1 to n
(3)     if (A[i] > A[j])
(4)       exchange A[i] and A[j]
```

Took a difficult problem:
Counting *all* comparisons
made by code

What did we really do?

```
(1) for i = 1 to n-1
(2)   for j = i+1 to n
(3)     if (A[i] > A[j])
(4)       exchange A[i] and A[j]
```

Took a difficult problem:

Counting *all* comparisons made by code

comparisons when $i = 1$:	$n - 1$
comparisons when $i = 2$:	$n - 2$
$\vdots$	
comparisons when $i = t$:	$n - t$
$\vdots$	
comparisons when $i = n - 1$:	1

Split into *simpler* parts

Easier to count in each part

Add parts together to get

$$1 + 2 + 3 + \ldots + (n - 1)$$

We showed how to *partition* a
large *set* of comparisons into
the *union* of smaller *mutually disjoint* sets.

We then derived our result using a general principle, the

(Sum Principle)

The size of a union of a
family of mutually disjoint finite sets
is the sum of sizes of the sets.

The next few slides are devoted
to defining all of the *red* terms.

Sets

Sets

Definition: *A set is a collection of objects*

Examples:

- The set of all men in this class
- The set of all women in this class
- The set of all students in this class surnamed *Ng*
- The set of all departments in the Engineering School
  $S = \{$ COMP, EEE, MechE, CivilE, ChemE, IE&EM $\}$

Sets

Definition: *A set is a collection of objects*

Examples:

- The set of all men in this class
- The set of all women in this class
- The set of all students in this class surnamed *Ng*
- The set of all departments in the Engineering School
  $S =$ { COMP, EEE, MechE, CivilE, ChemE, IE&EM }

Notation: Sets are usually denoted as $S = \{a, b, c\}$

In this case, set S contains *elements* or *objects* a, b and c

Definition: Two sets are **disjoint** if they have no elements in common.

Definition: Two sets are **disjoint** if they have no elements in common.

Examples:

- $S_1 = \{a, b, c\}$, $S_2 = \{a, e, f\}, S_3 = \{d, e, f\}$
- $S_4 =$ The set of all men in this class
- $S_5 =$ The set of all women in this class
- $S_6 =$ The set of all students in this class surnamed *Wong*
- $S_7 =$ The set of all students in this class surnamed *Chan*

Definition: Two sets are **disjoint** if they have no elements in common.

Examples:

- $S_1 = \{a, b, c\}$, $S_2 = \{a, e, f\}, S_3 = \{d, e, f\}$
- $S_4 =$ The set of all men in this class
- $S_5 =$ The set of all women in this class
- $S_6 =$ The set of all students in this class surnamed *Wong*
- $S_7 =$ The set of all students in this class surnamed *Chan*

Which sets are disjoint?

Definition: A set of sets $\{S_1, \ldots, S_m\}$
is a family of **mutually disjoint sets**,
if **every** pair of sets S_i, S_j in the family are disjoint.

Definition: A set of sets $\{S_1, \ldots, S_m\}$
is a family of **mutually disjoint sets**,
if **every** pair of sets S_i, S_j in the family are disjoint.

Example:

$S_1 = \{a, b, c\}$, $S_2 = \{d, e, f\}$, $S_3 = \{g, h, i\}$,
$S_4 = \{j, k, l\}$, $S_5 = \{k, l, m\}$

Which families are (not) mutually disjoint?

Definition: A set of sets $\{S_1, \ldots, S_m\}$
is a family of **mutually disjoint sets**,
if **every** pair of sets S_i, S_j in the family are disjoint.

Example:
$S_1 = \{a, b, c\}$, $S_2 = \{d, e, f\}$, $S_3 = \{g, h, i\}$,
$S_4 = \{j, k, l\}$, $S_5 = \{k, l, m\}$

Which families are (not) mutually disjoint?

- S_1, S_2, S_3, S_4 are mutually disjoint
- S_1, S_2, S_3, S_5 are mutually disjoint
- S_1, S_2, S_3, S_4, S_5 are not mutually disjoint

Definition: The size, $|S|$, of set S is the number of different items in S

Example:

If $S_1 = \{a, b, c\}$, $S_2 = \{a, b, c, d\}$,

Then $|S_1| = 3$, $|S_2| = 4$,

Definition: The size, $|S|$, of set S is the number of different items in S

Example:

If $S_1 = \{a, b, c\}$, $S_2 = \{a, b, c, d\}$,

Then $|S_1| = 3$, $|S_2| = 4$,

Note: An item can either be in or not in a set.

It can not be in a set more than once.

So, $S_3 = \{a, b, c, a\}$, denotes exactly the same set as $S_1 = \{a, b, c\}$, , i.e., $S_3 = S_1$ and $|S_3| = |S_1| = 3$

Definition: The union of sets $S_1, S_2, \ldots, S_m$ is the set S that contains exactly all of the objects that appear in at least one of the S_i

Definition: The union of sets $S_1, S_2, \ldots, S_m$ is the set S that contains exactly all of the objects that appear in at least one of the S_i

Example:

$S_1 = \{a, b, c\}$, $S_2 = \{d, e, f\}$, $S_3 = \{a, f, g, h\}$ $S_4 = \{g, h\}$

Definition: The union of sets $S_1, S_2, \ldots, S_m$ is the set S that contains exactly all of the objects that appear in at least one of the S_i

Example:

$S_1 = \{a, b, c\}$, $S_2 = \{d, e, f\}$, $S_3 = \{a, f, g, h\}$ $S_4 = \{g, h\}$

- $S_1 \cup S_2 = \{a, b, c, d, e, f\}$
- $S_1 \cup S_3 = \{a, b, c, f, g, h\}$
- $S_1 \cup S_2 \cup S_3 = \{a, b, c, d, e, f, g, h\}$
- $S_1 \cup S_2 \cup S_4 = \{a, b, c, d, e, f, g, h\}$

Definition: The union of sets $S_1, S_2, \ldots, S_m$ is the set S that contains exactly all of the objects that appear in at least one of the S_i

Example:
$S_1 = \{a, b, c\}$, $S_2 = \{d, e, f\}$, $S_3 = \{a, f, g, h\}$ $S_4 = \{g, h\}$

Definition:
$S_1, S_2, \ldots, S_m$ are a partition of $S = S_1 \cup S_2 \cup \cdots \cup S_m$
if $S_1, S_2, \ldots, S_m$ are a family of mutually disjoint sets
The S_i are the *blocks* of the partition

Definition: The union of sets $S_1, S_2, \ldots, S_m$ is the set S that contains exactly all of the objects that appear in at least one of the S_i

Example:

$S_1 = \{a, b, c\}$, $S_2 = \{d, e, f\}$, $S_3 = \{a, f, g, h\}$ $S_4 = \{g, h\}$

Definition:

$S_1, S_2, \ldots, S_m$ are a partition of $S = S_1 \cup S_2 \cup \cdots \cup S_m$ if $S_1, S_2, \ldots, S_m$ are a family of mutually disjoint sets

The S_i are the *blocks* of the partition

Example (cont):

$S = \{a, b, c, d, e, f, g, h\} = S_1 \cup S_2 \cup S_3 = S_1 \cup S_2 \cup S_4$

Definition: The union of sets $S_1, S_2, \ldots, S_m$ is the set S that contains exactly all of the objects that appear in at least one of the S_i

Example:
$S_1 = \{a, b, c\}$, $S_2 = \{d, e, f\}$, $S_3 = \{a, f, g, h\}$ $S_4 = \{g, h\}$

Definition:
$S_1, S_2, \ldots, S_m$ are a partition of $S = S_1 \cup S_2 \cup \cdots \cup S_m$
if $S_1, S_2, \ldots, S_m$ are a family of mutually disjoint sets
The S_i are the *blocks* of the partition

Example (cont):
$S = \{a, b, c, d, e, f, g, h\} = S_1 \cup S_2 \cup S_3 = S_1 \cup S_2 \cup S_4$

Is S_1, S_2, S_3 a partition of S? $\boxed{}$

Is S_1, S_2, S_4 a partition of S? $\boxed{}$

Definition: The union of sets $S_1, S_2, \ldots, S_m$ is the set S that contains exactly all of the objects that appear in at least one of the S_i

Example:
$S_1 = \{a, b, c\}$, $S_2 = \{d, e, f\}$, $S_3 = \{a, f, g, h\}$ $S_4 = \{g, h\}$

Definition:
$S_1, S_2, \ldots, S_m$ are a partition of $S = S_1 \cup S_2 \cup \cdots \cup S_m$
if $S_1, S_2, \ldots, S_m$ are a family of mutually disjoint sets
The S_i are the *blocks* of the partition

Example (cont):
$S = \{a, b, c, d, e, f, g, h\} = S_1 \cup S_2 \cup S_3 = S_1 \cup S_2 \cup S_4$

Is S_1, S_2, S_3 a partition of S? $\boxed{\textbf{No}}$

Is S_1, S_2, S_4 a partition of S? $\boxed{}$

Definition: The union of sets $S_1, S_2, \ldots, S_m$ is the set S that contains exactly all of the objects that appear in at least one of the S_i

Example:

$S_1 = \{a, b, c\}$, $S_2 = \{d, e, f\}$, $S_3 = \{a, f, g, h\}$ $S_4 = \{g, h\}$

Definition:

$S_1, S_2, \ldots, S_m$ are a partition of $S = S_1 \cup S_2 \cup \cdots \cup S_m$ if $S_1, S_2, \ldots, S_m$ are a family of mutually disjoint sets

The S_i are the *blocks* of the partition

Example (cont):

$S = \{a, b, c, d, e, f, g, h\} = S_1 \cup S_2 \cup S_3 = S_1 \cup S_2 \cup S_4$

Is S_1, S_2, S_3 a partition of S? **No**

Is S_1, S_2, S_4 a partition of S? **Yes**

(Sum Principle)
The size of the union of a family of mutually disjoint finite sets is the sum of sizes of the individual sets.

(Sum Principle)
The size of the union of a family of mutually disjoint finite sets is the sum of sizes of the individual sets.

Equivalently
If $S_1, S_2, \ldots, S_m$ are mutually disjoints sets, then

$$|S_1 \cup S_2 \cup \cdots \cup S_m| = |S_1| + |S_2| + \cdots + |S_m|.$$

Example:

$$S_1 = \{a, b, c\}, \; S_2 = \{d, e, f\}, \; S_3 = \{g, h\},$$

$$S = S_1 \cup S_2 \cup S_3 = \{a, b, c, d, e, f, g, h\}$$

$$|S| = 8 = 3 + 3 + 2 = |S_1| + |S_2| + |S_3|$$

(Sum Principle)
The size of the union of a family of mutually disjoint finite sets is the sum of sizes of the individual sets.

Equivalently
If $S_1, S_2, \ldots, S_m$ are mutually disjoints sets, then

$$|S_1 \cup S_2 \cup \cdots \cup S_m| = |S_1| + |S_2| + \cdots + |S_m|.$$

To avoid the dots, we sometimes write

$$\left|\bigcup_{i=1}^{m} S_i\right| = \sum_{i=1}^{m} |S_i|.$$

So Far

We counted comparisons in

```
(1) for i = 1 to n-1
(2)   for j = i+1 to n
(3)     if (A[i] > A[j])
(4)       exchange A[i] and A[j]
```

by noting that there are $n - t$ comps when $i = t$. Summing over t gives a total of $(n-1) + (n-2) + \ldots + 2 + 1$ comps.

So Far

We counted comparisons in

```
(1) for i = 1 to n-1
(2)   for j = i+1 to n
(3)     if (A[i] > A[j])
(4)       exchange A[i] and A[j]
```

by noting that there are $n-t$ comps when $i=t$. Summing over t gives a total of $(n-1)+(n-2)+\ldots+2+1$ comps.

In set notation, let S_t be the set of all comparisons $A[i]>A[j]$ made when $i=t$

Then the S_t are mutually disjoint with $|S_t|=n-t$

From sum principle, set $S=\bigcup_{i=1}^{n-1} S_i$ of all comparisons has size $|S|=\sum_{i=1}^{n-1}|S_i|=(n-1)+(n-2)+\ldots+2+1$

Abstraction

Abstraction

The process of figuring out a general principle that explains why a certain computation makes sense is an example of the mathematical process of **abstraction**.

Abstraction

The process of figuring out a general principle that explains why a certain computation makes sense is an example of the mathematical process of **abstraction**.

We counted the number of comparisons by

(i) abstracting the problem to a set problem,

(ii) partitioning up our original set (of all comparisons) into subsets

(iii) and then used the sum principle to get the final answer by summing up the sizes of the blocks of the partition.

Abstraction

The process of figuring out a general principle that explains why a certain computation makes sense is an example of the mathematical process of **abstraction**.

We counted the number of comparisons by

(i) abstracting the problem to a set problem,

(ii) partitioning up our original set (of all comparisons) into subsets

(iii) and then used the sum principle to get the final answer by summing up the sizes of the blocks of the partition.

The value of abstraction is that recognizing the abstract elements of a problem often helps us solve subsequent problems.

Summing Consecutive Integers

Summing Consecutive Integers

Find a simpler form for $(n-1)+(n-2)+\cdots+1$.

Summing Consecutive Integers

Find a simpler form for $(n-1)+(n-2)+\cdots+1$.

First rewrite as

$$\sum_{i=1}^{n-1}(n-i).$$

Summing Consecutive Integers

Find a simpler form for $(n-1)+(n-2)+\cdots+1$.

First rewrite as

$$\sum_{i=1}^{n-1}(n-i).$$

By reading from right to left instead of left to write we observe that

$$\sum_{i=1}^{n-1}(n-i)=\sum_{i=1}^{n-1} i.$$

Find a simpler form for $(n-1)+(n-2)+\cdots+1$.

Find a simpler form for $(n-1)+(n-2)+\cdots+1$.

Using a clever trick by Carl Friedrich Gauss

$$
\begin{array}{ccccccccccc}
 & 1 & + & 2 & + & \cdots & + & (n-2) & + & (n-1) \\
+ & (n-1) & + & (n-2) & + & \cdots & + & 2 & + & 1 \\
\hline
 & n & + & n & + & \cdots & + & n & + & n
\end{array}
$$

Find a simpler form for $(n-1)+(n-2)+\cdots+1$.

Using a clever trick by Carl Friedrich Gauss

$$\begin{array}{cccccccccccc}
 & 1 & + & 2 & + & \cdots & + & (n-2) & + & (n-1) \\
+ & (n-1) & + & (n-2) & + & \cdots & + & 2 & + & 1 \\
\hline
 & n & + & n & + & \cdots & + & n & + & n
\end{array}$$

Find a simpler form for $(n-1)+(n-2)+\cdots+1$.

Using a clever trick by Carl Friedrich Gauss

$$\begin{array}{cccccccccccc} & 1 & + & 2 & + & \cdots & + & (n-2) & + & (n-1) \\ + & (n-1) & + & (n-2) & + & \cdots & + & 2 & + & 1 \\ \hline & n & + & n & + & \cdots & + & n & + & n \end{array}$$

$(n-1)$ terms, each equal to n $\Rightarrow$ sum of two rows $= n(n-1)$.

Find a simpler form for $(n-1)+(n-2)+\cdots+1$.

Using a clever trick by Carl Friedrich Gauss

$$\begin{array}{cccccccccc}
 & 1 & + & 2 & + & \cdots & + & (n-2) & + & (n-1) \\
+ & (n-1) & + & (n-2) & + & \cdots & + & 2 & + & 1 \\
\hline
 & n & + & n & + & \cdots & + & n & + & n
\end{array}$$

$(n-1)$ terms, each equal to n $\Rightarrow$ sum of two rows $= n(n-1)$.

Both 1st and 2nd row have sum $\sum_{i=1}^{n-1} i$

$$\Rightarrow \; 2 * \sum_{i=1}^{n-1} i = n(n-1)$$

Find a simpler form for $(n-1)+(n-2)+\cdots+1$.

Using a clever trick by Carl Friedrich Gauss

$$\begin{array}{ccccccccccc} & 1 & + & 2 & + & \cdots & + & (n-2) & + & (n-1) \\ + & (n-1) & + & (n-2) & + & \cdots & + & 2 & + & 1 \\ \hline & n & + & n & + & \cdots & + & n & + & n \end{array}$$

$(n-1)$ terms, each equal to n $\Rightarrow$ sum of two rows $= n(n-1)$.

Both 1st and 2nd row have sum $\sum_{i=1}^{n-1} i$

$$\Rightarrow \; 2 * \sum_{i=1}^{n-1} i = n(n-1)$$

$$\Rightarrow \boxed{\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}}.$$

Find a simpler form for $(n-1)+(n-2)+\cdots+1$.

Using a clever trick by Carl Friedrich Gauss

Gauss on old German 10DM bill

Find a simpler form for $(n-1)+(n-2)+\cdots+1$.

Using a clever trick by Carl Friedrich Gauss

Gauss on old German 10DM bill

Sidenote : Gauss (1777-1855) was one of the most brilliant mathematicians in history. This particular trick was supposedly discovered by him during his first year of school (age 7).

An alternative derivation

We already saw that $\sum_{i=1}^{n-1} i = \sum_{i=1}^{n-1}(n-i)$ so

$$\begin{aligned} 2\sum_{i=1}^{n-1} i &= \sum_{i=1}^{n-1} i + \sum_{i=1}^{n-1}(n-i) \\ &= \sum_{i=1}^{n-1} [i + (n-i)] \\ &= \sum_{i=1}^{n-1} n = n(n-1) \end{aligned}$$

The Product Principle

The Product Principle

The following loop is a part of program computing the product of two matrices.

The Product Principle

The following loop is a part of program computing the product of two matrices.

```
for i = 1 to r
    for j = 1 to m
        S = 0
        for k = 1 to n
            S = S + A[i,k] * B[k,j]
        C[i,j] = S
```

The Product Principle

The following loop is a part of program computing the product of two matrices.

```
(1) for i = 1 to r
(2)   for j = 1 to m
(3)     S = 0
(4)     for k = 1 to n
(5)       S = S + A[i,k] * B[k,j]
(6)     C[i,j] = S
```

How many multiplications (in terms of r, m, n) does this pseudo-code carry out in total among all iterations of line 5?

```
for i = 1 to r
   for j = 1 to m
      S = 0
      for k = 1 to n
         S = S + A[i,k] * B[k,j]
      C[i,j] = S
```

```
(1) for i = 1 to r
(2)    for j = 1 to m
(3)       S = 0
(4)       for k = 1 to n
(5)          S = S + A[i,k] * B[k,j]
(6)       C[i,j] = S
```

Lines 4–5 ("**inner loop**") take exactly n steps.

```
(1) for i = 1 to r
(2)   for j = 1 to m
(3)     S = 0
(4)     for k = 1 to n
(5)       S = S + A[i,k] * B[k,j]
(6)     C[i,j] = S
```

Lines 4–5 ("**inner loop**") take exactly n steps.

Thus, it makes n multiplications,

regardless of values of variables i and j.

```
(1) for i = 1 to r
(2)     for j = 1 to m
(3)         S = 0
(4)         for k = 1 to n
(5)             S = S + A[i,k] * B[k,j]
(6)         C[i,j] = S
```

Lines 4–5 ("**inner loop**") take exactly n steps.

Thus, it makes n multiplications,
regardless of values of variables i and j.

Lines 2–5 repeat the inner loop exactly m times,
regardless of vaue of i.

```
(1) for i = 1 to r
(2)   for j = 1 to m
(3)     S = 0
(4)     for k = 1 to n
(5)       S = S + A[i,k] * B[k,j]
(6)     C[i,j] = S
```

Lines 4–5 ("**inner loop**") take exactly n steps.

Thus, it makes n multiplications,
regardless of values of variables i and j.

Lines 2–5 repeat the inner loop exactly m times,
regardless of vaue of i.

Therefore, this program segment makes
n multiplications m times $\Rightarrow nm$ multiplications.

Why multiply here?

Why multiply here?

Use abstract point of view:

Algorithm performs a certain *set* of multiplications.

Why multiply here?

Use abstract point of view:
Algorithm performs a certain *set* of multiplications.

For any fixed i, partition set of multiplications performed in lines 2–5 into:

set S_1 of multiplications performed when $j = 1$,
set S_2 of multiplications performed when $j = 2$,
$\vdots$
set S_t of multiplications performed when $j = t$
$\vdots$

Why multiply here?

Use abstract point of view:

Algorithm performs a certain *set* of multiplications.

For any fixed i, partition set of multiplications performed in lines 2–5 into:

set S_1 of multiplications performed when $j = 1$,

set S_2 of multiplications performed when $j = 2$,

$\vdots$

set S_t of multiplications performed when $j = t$

$\vdots$

Each S_j consists of multiplications of the inner loop for fixed j

Why multiply here?

Use abstract point of view:

Algorithm performs a certain *set* of multiplications.

For any fixed i, partition set of multiplications performed in lines 2–5 into:

set S_1 of multiplications performed when $j = 1$,

set S_2 of multiplications performed when $j = 2$,

$\vdots$

set S_t of multiplications performed when $j = t$

$\vdots$

Each S_j consists of multiplications of the inner loop for fixed j

and each set S_j contains exacty n multiplications

T_i: set of multiplications carried out for given i.

T_i: set of multiplications carried out for given i.

$$T_i = \bigcup_{j=1}^{m} S_j.$$

T_i: set of multiplications carried out for given i.

$$T_i = \bigcup_{j=1}^{m} S_j.$$

By sum principle,

$$T_i = |\bigcup_{j=1}^{m} S_j| = \sum_{j=1}^{m} |S_j| = \sum_{j=1}^{m} n = mn.$$

T_i: set of multiplications carried out for given i.

$$T_i = \bigcup_{j=1}^{m} S_j.$$

By sum principle,

$$T_i = |\bigcup_{j=1}^{m} S_j| = \sum_{j=1}^{m} |S_j| = \sum_{j=1}^{m} n = mn.$$

Thus, we multiplied because multiplication is repeated addition.

T_i: set of multiplications carried out for given i.

$$T_i = \bigcup_{j=1}^{m} S_j.$$

By sum principle,

$$T_i = |\bigcup_{j=1}^{m} S_j| = \sum_{j=1}^{m} |S_j| = \sum_{j=1}^{m} n = mn.$$

Thus, we multiplied because multiplication is repeated addition.

This yields the

(Product Principle) The size of the union of m disjoint sets, each of size n, is mn.

Back to the exercise:

Back to the exercise:

Lines 2–5 are executed once for each i from 1 to r.

Back to the exercise:

Lines 2–5 are executed once for each i from 1 to r.
i is different each time, so set of multiplications in one execution is disjoint from set of multiplications in any other.

Back to the exercise:

Lines 2–5 are executed once for each i from 1 to r. i is different each time, so set of multiplications in one execution is disjoint from set of multiplications in any other.

Thus, set of *all* multiplications is union of r disjoint sets T_i, each containing mn multiplications.

Back to the exercise:

Lines 2–5 are executed once for each i from 1 to r.
i is different each time, so set of multiplications in one execution is disjoint from set of multiplications in any other.

Thus, set of *all* multiplications is union of r disjoint sets T_i, each containing mn multiplications.

By *product principle*, set of *all* multiplications
has size rmn.

Back to the exercise:

Lines 2–5 are executed once for each i from 1 to r.
i is different each time, so set of multiplications in one execution is disjoint from set of multiplications in any other.

Thus, set of *all* multiplications is union of r disjoint sets T_i, each containing mn multiplications.

By *product principle*, set of *all* multiplications
has size rmn.
Then, program carries out, in total,
rmn multiplications

Two-Element Subsets

Two-Element Subsets

Example:

```
(1) for i = 1 to n-1
(2)    for j = i+ 1 to n
(3)       if (A[i] > A[j])
(4)          exchange A[i] and A[j]
```

Two-Element Subsets

Example:

```
(1) for i = 1 to n-1
(2)    for j = i+ 1 to n
(3)       if (A[i] > A[j])
(4)          exchange A[i] and A[j]
```

Before, we counted total no. of comparisons by partitioning set of comps into disjoint subsets and then using the *sum principle* to derive $n(n-1)/2$. We will now see a different way of counting the same thing

Two-Element Subsets

Example:

```
(1) for i = 1 to n-1
(2)    for j = i+ 1 to n
(3)       if (A[i] > A[j])
(4)          exchange A[i] and A[j]
```

Consider set of *comparisons*:

Two-Element Subsets

Example:

```
(1) for i = 1 to n-1
(2)    for j = i+ 1 to n
(3)       if (A[i] > A[j])
(4)          exchange A[i] and A[j]
```

Consider set of *comparisons*:

When $i = 1$, j takes on every value from 2 to n.

Two-Element Subsets

Example:

```
(1) for i = 1 to n-1
(2)    for j = i+ 1 to n
(3)       if (A[i] > A[j])
(4)          exchange A[i] and A[j]
```

Consider set of *comparisons*:

When $i = 1$, j takes on every value from 2 to n.
When $i = 2$, j takes on every value from 3 to n.

Two-Element Subsets

Example:

```
(1) for i = 1 to n-1
(2)    for j = i+ 1 to n
(3)       if (A[i] > A[j])
(4)          exchange A[i] and A[j]
```

Consider set of *comparisons*:

When $i = 1$, j takes on every value from 2 to n.
When $i = 2$, j takes on every value from 3 to n.

In general, for each pair of numbers i and j, compare $A[\mathbf{i}]$ and $A[j]$ exactly once.

Two-Element Subsets

Example:

```
(1) for i = 1 to n-1
(2)    for j = i+ 1 to n
(3)       if (A[i] > A[j])
(4)          exchange A[i] and A[j]
```

Consider set of *comparisons*:

When $i = 1$, j takes on every value from 2 to n.
When $i = 2$, j takes on every value from 3 to n.

In general, for each pair of numbers i and j, compare $A[\mathbf{i}]$ and $A[j]$ exactly once.

Thus, number of comparisons is the same as number of two-element subsets of $\{1, 2, \ldots, n\}$.

In how many ways can we choose an ordered pair from $\{1, 2, \ldots, n\}$?

Example: In *ordered pair*, $(2, 5)$ is different from $(5, 2)$.

In how many ways can we choose an
ordered pair from $\{1, 2, \ldots, n\}$?

Example: In *ordered pair*, $(2, 5)$ is different from $(5, 2)$.

n ways to choose a first element,
for each first choice, $n-1$ ways to choose a second element.

In how many ways can we choose an ordered pair from $\{1, 2, \ldots, n\}$?

Example: In *ordered pair*, $(2, 5)$ is different from $(5, 2)$.

n ways to choose a first element,
for each first choice, $n-1$ ways to choose a second element.

Thus, set of all choices is union of n sets of size $n-1$.
By product principle, this is $n(n-1)$

In how many ways can we choose an ordered pair from $\{1, 2, \ldots, n\}$?

Example: In *ordered pair*, $(2, 5)$ is different from $(5, 2)$.

n ways to choose a first element,
for each first choice, $n-1$ ways to choose a second element.

Thus, set of all choices is union of n sets of size $n-1$.
By product principle, this is $n(n-1)$

Each pair $\{a, b\}$ of distinct elements of $\{1, 2, \ldots, n\}$ can be ordered in two ways, (a, b) and (b, a).

So, there are twice as many ordered pairs as two-element sets.

In how many ways can we choose two elements from $\{1, 2, \ldots, n\}$?

Number of ordered pairs is $n(n-1)$,
so number of two-element subsets is $n(n-1)/2$.

In how many ways can we choose two elements from $\{1, 2, \ldots, n\}$?

Number of ordered pairs is $n(n-1)$,
so number of two-element subsets is $n(n-1)/2$.

$\binom{n}{2}$ denotes the number of two-element subsets of an n-element set. ("n choose 2")

In how many ways can we choose two elements from $\{1, 2, \ldots, n\}$?

Number of ordered pairs is $n(n-1)$, so number of two-element subsets is $n(n-1)/2$.

$\binom{n}{2}$ denotes the number of two-element subsets of an n-element set. ("n choose 2")

$$\boxed{\binom{n}{2} = \frac{n(n-1)}{2} = 1 + 2 + \cdots + (n-1)}$$

This is the end of Lecture 1.

In it we learnt some basic counting techniques (using set abstraction) and applied them to counting the number of comparisons in *selection-sort* and the number of ways to choose 2 items out of n.

These both turned out to be equal to

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

Please see section 1.1 of the book for more details