# Permutations, Combinations and the Binomial Theorem

October 27, 2011

# Permutation, revisited

## Definition

*An r-permutation from n distinct objects is an ordered selection of r objects from the given n objects.*

# Permutation, revisited

**Definition**

*An r-permutation from n distinct objects is an ordered selection of r objects from the given n objects.*

**Remark**

*By the product rule, there are* $n \cdot (n-1) \cdot \ldots \cdot (n-r+1)$ *different ways to orderly select r objects.*

# Permutation, revisited

## Definition

*An r-permutation from n distinct objects is an ordered selection of r objects from the given n objects.*

## Remark

*By the product rule, there are* $n \cdot (n-1) \cdot \ldots \cdot (n-r+1)$ *different ways to orderly select r objects.*

## Example

*There are 40 students in our class. In how many ways can we choose a class leader, a class organizer and a class treasurer to form the class committee?*

# Permutation, revisited

## Definition

*An r-permutation from n distinct objects is an ordered selection of r objects from the given n objects.*

## Remark

*By the product rule, there are* $n \cdot (n-1) \cdot \ldots \cdot (n-r+1)$ *different ways to orderly select r objects.*

## Example

*There are 40 students in our class. In how many ways can we choose a class leader, a class organizer and a class treasurer to form the class committee?*

## Answer

*This task can be performed in* $40 \cdot 39 \cdot 38$ *different ways.*

# Permutations and Sorting

# Permutations and Sorting

One of the most frequent activities of computers in large corporations is sorting. Needless to say that it is very important to devise sorting programs that will be as efficient as possible.

In many applications it is not unusual to have millions of records that need to be sorted.

We shall assume that a sorted sequence is a monotonically increasing sequence.

# Permutations and Sorting

One of the most frequent activities of computers in large corporations is sorting. Needless to say that it is very important to devise sorting programs that will be as efficient as possible.
In many applications it is not unusual to have millions of records that need to be sorted.
We shall assume that a sorted sequence is a monotonically increasing sequence.

**Definition**

*An inversion in a permutation $a_1 a_2 \ldots a_n$ is a pair of entries $a_i, a_j$ such that $i < j$, and $a_i > a_j$*

# Permutations and Sorting

One of the most frequent activities of computers in large corporations is sorting. Needless to say that it is very important to devise sorting programs that will be as efficient as possible.

In many applications it is not unusual to have millions of records that need to be sorted.

We shall assume that a sorted sequence is a monotonically increasing sequence.

**Definition**

*An inversion in a permutation $a_1 a_2 \ldots a_n$ is a pair of entries $a_i, a_j$ such that $i < j$, and $a_i > a_j$*

**Example (How many inversions are in these permutation?)**

# Permutations and Sorting

One of the most frequent activities of computers in large corporations is sorting. Needless to say that it is very important to devise sorting programs that will be as efficient as possible.
In many applications it is not unusual to have millions of records that need to be sorted.
We shall assume that a sorted sequence is a monotonically increasing sequence.

**Definition**

*An inversion in a permutation* $a_1 a_2 \dots a_n$ *is a pair of entries* $a_i, a_j$ *such that* $i < j, \ \ \text{and} \ \ a_i > a_j$

**Example (How many inversions are in these permutation?)**

1. 9 7 11 1 5 4 2 3 6 10 8 12

# Permutations and Sorting

One of the most frequent activities of computers in large corporations is sorting. Needless to say that it is very important to devise sorting programs that will be as efficient as possible.
In many applications it is not unusual to have millions of records that need to be sorted.
We shall assume that a sorted sequence is a monotonically increasing sequence.

**Definition**

*An inversion in a permutation* $a_1 a_2 \ldots a_n$ *is a pair of entries* $a_i, a_j$ *such that* $i < j, \;$ *and* $\; a_i > a_j$

**Example (How many inversions are in these permutation?)**

1. 9 7 11 1 5 4 2 3 6 10 8 12
2. 12 8 10 6 3 2 4 5 1 11 7 9

### Remark

*A sorted sequence (array) is a sequence with no inversions.*
*Thus the goal of a sorting procedure is to remove all inversions from the given sequence.*

### Remark

*A sorted sequence (array) is a sequence with no inversions. Thus the goal of a sorting procedure is to remove all inversions from the given sequence.*

### Question

*What is the average number of inversions in an n-permutation?*

### Remark

*A sorted sequence (array) is a sequence with no inversions.*
*Thus the goal of a sorting procedure is to remove all inversions from the given sequence.*

### Question

*What is the average number of inversions in an n-permutation?*

### Answer

### Remark

*A sorted sequence (array) is a sequence with no inversions.*
*Thus the goal of a sorting procedure is to remove all inversions from the given sequence.*

### Question

*What is the average number of inversions in an n-permutation?*

### Answer

1. *There are* $n!$ *distinct permutations.*

## Remark

*A sorted sequence (array) is a sequence with no inversions.*
*Thus the goal of a sorting procedure is to remove all inversions from the given sequence.*

## Question

*What is the average number of inversions in an n-permutation?*

## Answer

1. *There are $n!$ distinct permutations.*
2. *A permutation can have $0$ inversions (sorted) or $\binom{n}{2}$ inversions or any number in between.*

## Remark

*A sorted sequence (array) is a sequence with no inversions.*
*Thus the goal of a sorting procedure is to remove all inversions from the given sequence.*

## Question

*What is the average number of inversions in an n-permutation?*

## Answer

1. *There are $n!$ distinct permutations.*
2. *A permutation can have $0$ inversions (sorted) or $\binom{n}{2}$ inversions or any number in between.*
3. *The average number of inversions in a random permutation is the total number of inversions in all $n!$ permutations divided by $n!$.*

## Remark

*A sorted sequence (array) is a sequence with no inversions.*
*Thus the goal of a sorting procedure is to remove all inversions from the given sequence.*

## Question

*What is the average number of inversions in an n-permutation?*

## Answer

1. *There are* $n!$ *distinct permutations.*
2. *A permutation can have* $0$ *inversions (sorted) or* $\binom{n}{2}$ *inversions or any number in between.*
3. *The average number of inversions in a random permutation is the total number of inversions in all* $n!$ *permutations divided by* $n!$.
4. *But how can we find the total number?*

1. We shall count the total number of inversions in pairs.

1. We shall count the total number of inversions in pairs.
2. We pair every permutation $a_1 a_2 \ldots a_{n-1} a_n$ with its reverse $a_n a_{n-1} \ldots a_2 a_1$.

1. We shall count the total number of inversions in pairs.
2. We pair every permutation $a_1 a_2 \ldots a_{n-1} a_n$ with its reverse $a_n a_{n-1} \ldots a_2 a_1$.
3. We have $\frac{n!}{2}$ disjoint pairs.

1. We shall count the total number of inversions in pairs.
2. We pair every permutation $a_1 a_2 \ldots a_{n-1} a_n$ with its reverse $a_n a_{n-1} \ldots a_2 a_1$.
3. We have $\frac{n!}{2}$ disjoint pairs.
4. Each pair accounts for $\binom{n}{2}$ inversions.

1. We shall count the total number of inversions in pairs.
2. We pair every permutation $a_1 a_2 \ldots a_{n-1} a_n$ with its reverse $a_n a_{n-1} \ldots a_2 a_1$.
3. We have $\frac{n!}{2}$ disjoint pairs.
4. Each pair accounts for $\binom{n}{2}$ inversions.
5. So the average number of inversions in an $n$-permutation is: $\frac{1}{n!}\binom{n}{2} \cdot \frac{n!}{2} = \frac{n(n-1)}{4}$

1. We shall count the total number of inversions in pairs.
2. We pair every permutation $a_1 a_2 \ldots a_{n-1} a_n$ with its reverse $a_n a_{n-1} \ldots a_2 a_1$.
3. We have $\frac{n!}{2}$ disjoint pairs.
4. Each pair accounts for $\binom{n}{2}$ inversions.
5. So the average number of inversions in an $n$-permutation is: $\frac{1}{n!}\binom{n}{2} \cdot \frac{n!}{2} = \frac{n(n-1)}{4}$

**Remark**

*If we exchange $a_i$ and $a_{i+1}$ we remove 1 inversion.*

1. We shall count the total number of inversions in pairs.
2. We pair every permutation $a_1 a_2 \ldots a_{n-1} a_n$ with its reverse $a_n a_{n-1} \ldots a_2 a_1$.
3. We have $\frac{n!}{2}$ disjoint pairs.
4. Each pair accounts for $\binom{n}{2}$ inversions.
5. So the average number of inversions in an $n$-permutation is: $\frac{1}{n!}\binom{n}{2} \cdot \frac{n!}{2} = \frac{n(n-1)}{4}$

**Remark**

*If we exchange $a_i$ and $a_{i+1}$ we remove 1 inversion.*

1. We shall count the total number of inversions in pairs.
2. We pair every permutation $a_1 a_2 \ldots a_{n-1} a_n$ with its reverse $a_n a_{n-1} \ldots a_2 a_1$.
3. We have $\frac{n!}{2}$ disjoint pairs.
4. Each pair accounts for $\binom{n}{2}$ inversions.
5. So the average number of inversions in an $n$-permutation is: $\frac{1}{n!}\binom{n}{2} \cdot \frac{n!}{2} = \frac{n(n-1)}{4}$

### Remark

*If we exchange $a_i$ and $a_{i+1}$ we remove 1 inversion.*

*So on the average, we'll have to perform $\frac{n(n-1)}{4}$ such exchanges.*

*Better sorting programs compare records that are far apart thus capable of removing more inversions in one exchange.*

# Sorting

Let us try to see what is the most efficient execution for sorting 5 objects. The best model for analyzing this problem seems to be the decision tree model.

# Sorting

Let us try to see what is the most efficient execution for sorting 5 objects. The best model for analyzing this problem seems to be the decision tree model.
The decision tree for this problem (a binary tree) will have to have 5! leaves.

# Sorting

Let us try to see what is the most efficient execution for sorting 5 objects. The best model for analyzing this problem seems to be the decision tree model.

The decision tree for this problem (a binary tree) will have to have 5! leaves.

This means that its depth will have to be at least 7.

### Question

*Can we design a sorting algorithm that will sort any given 5 objects in no more than 7 comparisons?*

# Sorting

Let us try to see what is the most efficient execution for sorting 5 objects. The best model for analyzing this problem seems to be the decision tree model.

The decision tree for this problem (a binary tree) will have to have 5! leaves.

This means that its depth will have to be at least 7.

**Question**

*Can we design a sorting algorithm that will sort any given* 5 *objects in no more than* 7 *comparisons?*

**Question**

*For a fixed integer $n$ what is the smallest number of comparisons a sorting algorithm needs to execute to sort any input list of $n$ objects?*

1. The decision tree model for analyzing sorting will have to be a tree with $n!$ leaves.

1. The decision tree model for analyzing sorting will have to be a tree with $n!$ leaves.
2. This means that the height of the tree is $\geq \lceil \log n! \rceil$.

1. The decision tree model for analyzing sorting will have to be a tree with $n!$ leaves.
2. This means that the height of the tree is $\geq \lceil \log n! \rceil$.
3. $$\log n! = \sum_{k=1}^{n} \log k \leq n \log n$$

1. The decision tree model for analyzing sorting will have to be a tree with $n!$ leaves.
2. This means that the height of the tree is $\geq \lceil \log n! \rceil$.
3. $$\log n! = \sum_{k=1}^{n} \log k \leq n \log n$$
4. $$\log n! > \sum_{k>\frac{n}{2}}^{n} \log k > \frac{1}{4} n \log n$$

1. The decision tree model for analyzing sorting will have to be a tree with $n!$ leaves.
2. This means that the height of the tree is $\geq \lceil \log n! \rceil$.
3. $$\log n! = \sum_{k=1}^{n} \log k \leq n \log n$$
4. $$\log n! > \sum_{k>\frac{n}{2}}^{n} \log k > \frac{1}{4} n \log n$$

1. The decision tree model for analyzing sorting will have to be a tree with $n!$ leaves.
2. This means that the height of the tree is $\geq \lceil \log n! \rceil$.
3. $$\log n! = \sum_{k=1}^{n} \log k \leq n \log n$$
4. $$\log n! > \sum_{k>\frac{n}{2}}^{n} \log k > \frac{1}{4} n \log n$$

We do have sorting algorithms that execute about $c \cdot n \log n$ comparisons.

# Enumerating Permutations

In many applications, for instance if we need to generate random permutations we need to enumerate permutations.

# Enumerating Permutations

In many applications, for instance if we need to generate random permutations we need to enumerate permutations.

That is we need to find a bijection $f : S_n \to \{0, 1, \ldots (n! - 1)\}$.

# Enumerating Permutations

In many applications, for instance if we need to generate random permutations we need to enumerate permutations.

That is we need to find a bijection $f : S_n \to \{0, 1, \ldots (n! - 1)\}$.

# Cantor Digits

There are many different representations of integers:

1. In general, given a sequence $\alpha = a_1, a_2, \ldots$
   An $\alpha$ representation of the integer $n$ is:

# Cantor Digits

There are many different representations of integers:

1. In general, given a sequence $\alpha = a_1, a_2, \ldots$
   An $\alpha$ representation of the integer $n$ is:
   - $n = \sum_{i=0}^{m} d_i \cdot a_i$.

# Cantor Digits

There are many different representations of integers:

1. In general, given a sequence $\alpha = a_1, a_2, \ldots$
   An $\alpha$ representation of the integer $n$ is:
   - $n = \sum_{i=0}^{m} d_i \cdot a_i$.
   - There are restrictions on the coefficients $d_i$.

# Cantor Digits

There are many different representations of integers:

1. In general, given a sequence $\alpha = a_1, a_2, \ldots$
   An $\alpha$ representation of the integer $n$ is:
   - $n = \sum_{i=0}^{m} d_i \cdot a_i$.
   - There are restrictions on the coefficients $d_i$.
   - Every integer $n$ has such a representation.

# Cantor Digits

There are many different representations of integers:

1. In general, given a sequence $\alpha = a_1, a_2, \ldots$
   An $\alpha$ representation of the integer $n$ is:
   - $n = \sum_{i=0}^{m} d_i \cdot a_i$.
   - There are restrictions on the coefficients $d_i$.
   - Every integer $n$ has such a representation.
   - The representation is unique.

# Cantor Digits

There are many different representations of integers:

1. In general, given a sequence $\alpha = a_1, a_2, \ldots$
   An $\alpha$ representation of the integer $n$ is:
   - $n = \sum_{i=0}^{m} d_i \cdot a_i$.
   - There are restrictions on the coefficients $d_i$.
   - Every integer $n$ has such a representation.
   - The representation is unique.
2. Examples:

# Cantor Digits

There are many different representations of integers:

1. In general, given a sequence $\alpha = a_1, a_2, \ldots$
   An $\alpha$ representation of the integer $n$ is:
   - $n = \sum_{i=0}^{m} d_i \cdot a_i$.
   - There are restrictions on the coefficients $d_i$.
   - Every integer $n$ has such a representation.
   - The representation is unique.
2. Examples:
3. Decimal (common) $n = \sum_{k=0}^{m} a_k \cdot 10^k \quad 0 \leq a_k \leq 9$

# Cantor Digits

There are many different representations of integers:

1. In general, given a sequence $\alpha = a_1, a_2, \ldots$
   An $\alpha$ representation of the integer $n$ is:
   - $n = \sum_{i=0}^{m} d_i \cdot a_i$.
   - There are restrictions on the coefficients $d_i$.
   - Every integer $n$ has such a representation.
   - The representation is unique.
2. Examples:
3. Decimal (common) $n = \sum_{k=0}^{m} a_k \cdot 10^k \quad 0 \leq a_k \leq 9$
4. Example: $150436 = 6 \cdot 10^0 + 3 \cdot 10^1 + 4 \cdot 10^2 + 5 \cdot 10^4 + 1 \cdot 10^5$

# Cantor Digits

There are many different representations of integers:

1. In general, given a sequence $\alpha = a_1, a_2, \ldots$
   An $\alpha$ representation of the integer $n$ is:
   - $n = \sum_{i=0}^{m} d_i \cdot a_i$.
   - There are restrictions on the coefficients $d_i$.
   - Every integer $n$ has such a representation.
   - The representation is unique.
2. Examples:
3. Decimal (common) $n = \sum_{k=0}^{m} a_k \cdot 10^k \quad 0 \leq a_k \leq 9$
4. Example: $150436 = 6 \cdot 10^0 + 3 \cdot 10^1 + 4 \cdot 10^2 + 5 \cdot 10^4 + 1 \cdot 10^5$
5. Binary representation: $n = \sum_{i=0}^{m} d_i \cdot 2^i, \quad d_i = 0, 1$.

# Cantor Digits

There are many different representations of integers:

1. In general, given a sequence $\alpha = a_1, a_2, \ldots$
   An $\alpha$ representation of the integer $n$ is:
   - $n = \sum_{i=0}^{m} d_i \cdot a_i$.
   - There are restrictions on the coefficients $d_i$.
   - Every integer $n$ has such a representation.
   - The representation is unique.
2. Examples:
3. Decimal (common) $n = \sum_{k=0}^{m} a_k \cdot 10^k \quad 0 \leq a_k \leq 9$
4. Example: $150436 = 6 \cdot 10^0 + 3 \cdot 10^1 + 4 \cdot 10^2 + 5 \cdot 10^4 + 1 \cdot 10^5$
5. Binary representation: $n = \sum_{i=0}^{m} d_i \cdot 2^i, \quad d_i = 0, 1$.
6. Base $b$ representation: $n = \sum_{i=1}^{m} d_i \cdot b^i \quad 0 \leq d_i < b$.

# Cantor Digits

There are many different representations of integers:

1. In general, given a sequence $\alpha = a_1, a_2, \dots$
   An $\alpha$ representation of the integer $n$ is:
   - $n = \sum_{i=0}^{m} d_i \cdot a_i$.
   - There are restrictions on the coefficients $d_i$.
   - Every integer $n$ has such a representation.
   - The representation is unique.
2. Examples:
3. Decimal (common) $n = \sum_{k=0}^{m} a_k \cdot 10^k \quad 0 \le a_k \le 9$
4. Example: $150436 = 6 \cdot 10^0 + 3 \cdot 10^1 + 4 \cdot 10^2 + 5 \cdot 10^4 + 1 \cdot 10^5$
5. Binary representation: $n = \sum_{i=0}^{m} d_i \cdot 2^i, \quad d_i = 0, 1$.
6. Base $b$ representation: $n = \sum_{i=1}^{m} d_i \cdot b^i \quad 0 \le d_i < b$.
7. Cantor Digits: $n = \sum_{k=0}^{m} d_k \cdot k! \quad 0 \le d_k \le k$.

# Cantor Digits

## Example

# Cantor Digits

## Example

1. $1000 = 1 \cdot 6! + 2 \cdot 5! + 1 \cdot 4! + 2 \cdot 3! + 2 \cdot 2!$

# Cantor Digits

## Example

1. $1000 = 1 \cdot 6! + 2 \cdot 5! + 1 \cdot 4! + 2 \cdot 3! + 2 \cdot 2!$
2. *So the cantor digits of 1000 are* 1 2 1 2 2 0.

# Cantor Digits

## Example

1. $1000 = 1 \cdot 6! + 2 \cdot 5! + 1 \cdot 4! + 2 \cdot 3! + 2 \cdot 2!$
2. *So the cantor digits of 1000 are* 1 2 1 2 2 0*.*

## Remark

*For an $\alpha$-representation to be unique it is sufficient that $a_{n+1} > \sum_{i=0}^{n} d_i \cdot a_i$ for all possible choices of $d_i$.*

# Cantor Digits

## Example

1. $1000 = 1 \cdot 6! + 2 \cdot 5! + 1 \cdot 4! + 2 \cdot 3! + 2 \cdot 2!$
2. *So the cantor digits of 1000 are* 1 2 1 2 2 0*.*

## Remark

*For an $\alpha$-representation to be unique it is sufficient that $a_{n+1} > \sum_{i=0}^{n} d_i \cdot a_i$ for all possible choices of $d_i$.*

# Cantor Digits

## Example

1. $1000 = 1 \cdot 6! + 2 \cdot 5! + 1 \cdot 4! + 2 \cdot 3! + 2 \cdot 2!$
2. *So the cantor digits of 1000 are* 1 2 1 2 2 0*.*

## Remark

*For an $\alpha$-representation to be unique it is sufficient that $a_{n+1} > \sum_{i=0}^{n} d_i \cdot a_i$ for all possible choices of $d_i$.*

We need to show that :

# Cantor Digits

## Example

1. $1000 = 1 \cdot 6! + 2 \cdot 5! + 1 \cdot 4! + 2 \cdot 3! + 2 \cdot 2!$
2. *So the cantor digits of 1000 are* 1 2 1 2 2 0.

## Remark

*For an $\alpha$-representation to be unique it is sufficient that $a_{n+1} > \sum_{i=0}^{n} d_i \cdot a_i$ for all possible choices of $d_i$.*

We need to show that :

## Theorem

*Every integer m has a unique representation:*
$m = \sum_{k=0}^{s} d_k \cdot k! \quad 0 \leq d_k \leq k.$

Proof.

First recall that $\sum_{k=1}^{s} k \cdot k! = (s+1)! - 1$ so by the previous remark the representation is unique.

We now proceed by induction to prove that every integer has a Cantor Digits representation.

□

## Proof.

First recall that $\sum_{k=1}^{s} k \cdot k! = (s+1)! - 1$ so by the previous remark the representation is unique.

We now proceed by induction to prove that every integer has a Cantor Digits representation.

1. $1 = 1 \cdot 1!$

□

Proof.

First recall that $\sum_{k=1}^{s} k \cdot k! = (s+1)! - 1$ so by the previous remark the representation is unique.

We now proceed by induction to prove that every integer has a Cantor Digits representation.

1. $1 = 1 \cdot 1!$
2. Assume $m = \sum_{k=0}^{s} d_k \cdot k! \quad 0 \leq d_k \leq k$.
   We need to show that $m + 1 = \sum_{k=0}^{s} f_k \cdot k! \quad 0 \leq f_k \leq k$.

Proof.

First recall that $\sum_{k=1}^{s} k \cdot k! = (s+1)! - 1$ so by the previous remark the representation is unique.

We now proceed by induction to prove that every integer has a Cantor Digits representation.

1. $1 = 1 \cdot 1!$
2. Assume $m = \sum_{k=0}^{s} d_k \cdot k! \quad 0 \leq d_k \leq k$.
   We need to show that $m+1 = \sum_{k=0}^{s} f_k \cdot k! \quad 0 \leq f_k \leq k$.
3. If $d_k = k \; \forall k$ then $m = (s+1)! - 1$ *and* $m+1 = (s+1)!$.

□

## Proof.

First recall that $\sum_{k=1}^{s} k \cdot k! = (s+1)! - 1$ so by the previous remark the representation is unique.

We now proceed by induction to prove that every integer has a Cantor Digits representation.

1. $1 = 1 \cdot 1!$
2. Assume $m = \sum_{k=0}^{s} d_k \cdot k! \quad 0 \leq d_k \leq k$.
   We need to show that $m + 1 = \sum_{k=0}^{s} f_k \cdot k! \quad 0 \leq f_k \leq k$.
3. If $d_k = k \; \forall k$ then $m = (s+1)! - 1$ *and* $m + 1 = (s+1)!$.
4. Let $k$ be the smallest index for which $d_k < k$ (such an index exists).

## Proof.

First recall that $\sum_{k=1}^{s} k \cdot k! = (s+1)! - 1$ so by the previous remark the representation is unique.

We now proceed by induction to prove that every integer has a Cantor Digits representation.

1. $1 = 1 \cdot 1!$
2. Assume $m = \sum_{k=0}^{s} d_k \cdot k! \quad 0 \leq d_k \leq k$.
   We need to show that $m + 1 = \sum_{k=0}^{s} f_k \cdot k! \quad 0 \leq f_k \leq k$.
3. If $d_k = k \; \forall k$ then $m = (s+1)! - 1$ *and* $m + 1 = (s+1)!$.
4. Let $k$ be the smallest index for which $d_k < k$ (such an index exists).
5. That means that
   $m = 1 \cdot 1! + 2 \cdot 2! + \ldots + (k-1) \cdot (k-1)! + d_k \cdot k! + \ldots$

□

## Proof.

First recall that $\sum_{k=1}^{s} k \cdot k! = (s+1)! - 1$ so by the previous remark the representation is unique.

We now proceed by induction to prove that every integer has a Cantor Digits representation.

1. $1 = 1 \cdot 1!$
2. Assume $m = \sum_{k=0}^{s} d_k \cdot k! \quad 0 \leq d_k \leq k$.
   We need to show that $m + 1 = \sum_{k=0}^{s} f_k \cdot k! \quad 0 \leq f_k \leq k$.
3. If $d_k = k \; \forall k$ then $m = (s+1)! - 1$ *and* $m + 1 = (s+1)!$.
4. Let $k$ be the smallest index for which $d_k < k$ (such an index exists).
5. That means that
   $m = 1 \cdot 1! + 2 \cdot 2! + \ldots + (k-1) \cdot (k-1)! + d_k \cdot k! + \ldots$
6. $m + 1 = (d_k + 1) \cdot k! + \ldots$.

□

# Enumerating Permutations

Given an n- permutation $\pi = a_1 a_2 \ldots a_n$ we asociate with it the integer $f(\pi) = \sum_{k=1}^{n-1} d_k \cdot k!$.

The coefficients $d_k$ are calculated as follows:

Let $a_j = k + 1$. Then $d_k = |\{a_{i_m} | i_m > j \text{ and } (k+1) = a_j > a_{i_m}\}|$

In words: $d_k$ is the number of entries in the permutation $\pi$ that are to the right of $k+1$ and are smaller than $k+1$.

# Enumerating Permutations

Given an n- permutation $\pi = a_1 a_2 \ldots a_n$ we asociate with it the integer $f(\pi) = \sum_{k=1}^{n-1} d_k \cdot k!$.

The coefficients $d_k$ are calculated as follows:

Let $a_j = k + 1$. Then $d_k = |\{a_{i_m} | i_m > j \ and \ (k+1) = a_j > a_{i_m}\}|$

In words: $d_k$ is the number of entries in the permutation $\pi$ that are to the right of $k+1$ and are smaller than $k+1$.

**Example**

*Let* $\pi = 7\ 5\ 4\ 6\ 1\ 3\ 2\ 8.$

$d_1 = 0,\ d_2 = 1,\ d_3 = 3,\ d_4 = 4, d_5 = 3, d_6 = 6.$
*So* $\quad f(\pi) = 6 \cdot 6! + 3 \cdot 5! + 4 \cdot 4! + 3 \cdot 3! + 2!$

## Example

*Let us calculate the 8-permutation number* $20,000$.

## Example

*Let us calculate the 8-permutation number* $20,000$.

### Example

*Let us calculate the 8-permutation number* $20,000$.

$20000 = 2! + 3! + 3 * 4! + 4 * 5! + 6 * 6! + 3 * 7!$
*(use a simple greedy approach to make this easy calculation).*

### Example

*Let us calculate the 8-permutation number* $20,000$*.*

$20000 = 2! + 3! + 3 * 4! + 4 * 5! + 6 * 6! + 3 * 7!$
*(use a simple greedy approach to make this easy calculation).*

1. *Start with eight* $********$*. Each* $*$ *will represent one of the integers* $1, 2, \ldots 8$*.*

### Example

*Let us calculate the 8-permutation number* $20,000$.

$20000 = 2! + 3! + 3 * 4! + 4 * 5! + 6 * 6! + 3 * 7!$
*(use a simple greedy approach to make this easy calculation).*

1. *Start with eight* $* * * * * * **$. *Each* $*$ *will represent one of the integers* $1, 2, \ldots 8$.
2. $d_7 = 3,$ *so* $8$ *has* $3$ *smaller numbers following it.*
   *Place it so that* $3 * s$ *follow it:* $* * * * 8 * **$.

## Example

*Let us calculate the 8-permutation number* $20,000$.

$20000 = 2! + 3! + 3 * 4! + 4 * 5! + 6 * 6! + 3 * 7!$
*(use a simple greedy approach to make this easy calculation).*

1. *Start with eight* $********$. *Each* $*$ *will represent one of the integers* $1, 2, \ldots 8$.
2. $d_7 = 3,$ *so* $8$ *has* $3$ *smaller numbers following it.*
   *Place it so that* $3 *$ *s follow it:* $****8***$.
3. *Next place* $7$ *so that* $6\ *$ *s follow it* : $7***8***$

## Example

*Let us calculate the 8-permutation number* $20,000$.

$20000 = 2! + 3! + 3 * 4! + 4 * 5! + 6 * 6! + 3 * 7!$
*(use a simple greedy approach to make this easy calculation).*

1. *Start with eight* $********$. *Each* $*$ *will represent one of the integers* $1, 2, \ldots 8$.
2. $d_7 = 3,$ *so* $8$ *has* $3$ *smaller numbers following it.*
   *Place it so that* $3 *$ *s follow it:* $****8***$.
3. *Next place* $7$ *so that* $6$ $*$ *s follow it* : $7***8***$
4. *Place* $i+1$ *so that* $d_i *$ *s follow it.*

## Example

*Let us calculate the 8-permutation number* $20,000$.

$20000 = 2! + 3! + 3*4! + 4*5! + 6*6! + 3*7!$
*(use a simple greedy approach to make this easy calculation).*

1. *Start with eight* $********$. *Each* $*$ *will represent one of the integers* $1, 2, \ldots 8$.
2. $d_7 = 3,$ *so* $8$ *has* $3$ *smaller numbers following it.*
   *Place it so that* $3 *$ *s follow it:* $\quad ****8***$.
3. *Next place* $7$ *so that* $6 \; *$ *s follow it* : $\quad 7***8***$
4. *Place* $i+1$ *so that* $d_i *$ *s follow it.*
5. *place* $1$ *at the last* $*$.

## Example

*Let us calculate the 8-permutation number* $20,000$.

$20000 = 2! + 3! + 3 * 4! + 4 * 5! + 6 * 6! + 3 * 7!$
*(use a simple greedy approach to make this easy calculation).*

1. *Start with eight* $* * * * * * **$. *Each* $*$ *will represent one of the integers* $1, 2, \ldots 8$.
2. $d_7 = 3,$ *so* $8$ *has* $3$ *smaller numbers following it.*
   *Place it so that* $3 *$ *s follow it:* $* * * * 8 * **$.
3. *Next place* $7$ *so that* $6$ $*$ *s follow it* : $7 * * * 8 * **$
4. *Place* $i + 1$ *so that* $d_i *$ *s follow it.*
5. *place* $1$ *at the last* $*$.
6. *In our example:* $f^{-1}(20000) = 7\ 1\ 6\ 5\ 8\ 3\ 4\ 2$.

# Efficient Generation of Permutations and Combinations

Permutations can be generated either by the lexicographic order or by the Cantor-Digits enumeration.
There is another method called *The Arrow* algorithm.

1. Start by placing an arrow pointing to the left over each number in the $n$- permutation: $\overleftarrow{1}\ \overleftarrow{2} \ldots \overleftarrow{n}$.

# Efficient Generation of Permutations and Combinations

Permutations can be generated either by the lexicographic order or by the Cantor-Digits enumeration.
There is another method called *The Arrow* algorithm.

1. Start by placing an arrow pointing to the left over each number in the $n$- permutation: $\overleftarrow{1}\ \overleftarrow{2} \ldots \overleftarrow{n}$.
2. The next permutation is generated by finding the largest entry whose arrow points to a smaller entry then:

# Efficient Generation of Permutations and Combinations

Permutations can be generated either by the lexicographic order or by the Cantor-Digits enumeration.
There is another method called *The Arrow* algorithm.

1. Start by placing an arrow pointing to the left over each number in the $n$- permutation: $\overleftarrow{1}\ \overleftarrow{2} \ldots \overleftarrow{n}$.
2. The next permutation is generated by finding the largest entry whose arrow points to a smaller entry then:
    - Interchange the two numbers.

# Efficient Generation of Permutations and Combinations

Permutations can be generated either by the lexicographic order or by the Cantor-Digits enumeration.
There is another method called *The Arrow* algorithm.

1. Start by placing an arrow pointing to the left over each number in the $n$- permutation: $\overleftarrow{1}\ \overleftarrow{2} \ldots \overleftarrow{n}$.
2. The next permutation is generated by finding the largest entry whose arrow points to a smaller entry then:
   - Interchange the two numbers.
   - Reverse the direction of all arrows on numbers greater this entry.

# Efficient Generation of Permutations and Combinations

Permutations can be generated either by the lexicographic order or by the Cantor-Digits enumeration.
There is another method called *The Arrow* algorithm.

1. Start by placing an arrow pointing to the left over each number in the $n$- permutation: $\overleftarrow{1}\ \overleftarrow{2} \ldots \overleftarrow{n}$.
2. The next permutation is generated by finding the largest entry whose arrow points to a smaller entry then:
   - Interchange the two numbers.
   - Reverse the direction of all arrows on numbers greater this entry.
3. Stop when no arrow above an entry points to a smaller entry.

## Example

*Start:*

$$\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3}$$

### Example

*Start:*

$$\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3}$$

## Example

*Start:*

$$\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3} \Rightarrow \overleftarrow{1}\ \overleftarrow{3}\ \overleftarrow{2}$$

### Example

*Start:*

$$\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3} \Rightarrow \overleftarrow{1}\ \overleftarrow{3}\ \overleftarrow{2} \Rightarrow \overleftarrow{3}\ \overleftarrow{1}\ \overleftarrow{2}$$

## Example

*Start:*

$$\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3} \Rightarrow \overleftarrow{1}\ \overleftarrow{3}\ \overleftarrow{2} \Rightarrow \overleftarrow{3}\ \overleftarrow{1}\ \overleftarrow{2} \Rightarrow \overrightarrow{3}\ \overleftarrow{2}\ \overleftarrow{1}$$

### Example

*Start:*

$$\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3} \Rightarrow \overleftarrow{1}\ \overleftarrow{3}\ \overleftarrow{2} \Rightarrow \overleftarrow{3}\ \overleftarrow{1}\ \overleftarrow{2} \Rightarrow \overrightarrow{3}\ \overleftarrow{2}\ \overleftarrow{1} \Rightarrow \overleftarrow{2}\ \overrightarrow{3}\ \overleftarrow{1}$$

## Example

*Start:*

$$\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3} \Rightarrow \overleftarrow{1}\ \overleftarrow{3}\ \overleftarrow{2} \Rightarrow \overleftarrow{3}\ \overleftarrow{1}\ \overleftarrow{2} \Rightarrow \overrightarrow{3}\ \overleftarrow{2}\ \overleftarrow{1} \Rightarrow \overleftarrow{2}\ \overrightarrow{3}\ \overleftarrow{1} \Rightarrow \overleftarrow{2}\ \overleftarrow{1}\ \overrightarrow{3}$$

## Example

*Start:*

$$\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3} \Rightarrow \overleftarrow{1}\ \overleftarrow{3}\ \overleftarrow{2} \Rightarrow \overleftarrow{3}\ \overleftarrow{1}\ \overleftarrow{2} \Rightarrow \overrightarrow{3}\ \overleftarrow{2}\ \overleftarrow{1} \Rightarrow \overleftarrow{2}\ \overrightarrow{3}\ \overleftarrow{1} \Rightarrow \overleftarrow{2}\ \overleftarrow{1}\ \overrightarrow{3}$$

## Remark (Generating Combinations)

*We wish to generate all r-combinations of an n-set* $\{a_1, a_2, \ldots, a_n\}$. *We shall proceed lexicographically:* $\{a_1, a_2, \ldots a_r\}$ *will be the first ("smallest") and* $\{a_{n-r+1}, \ldots, a_n\}$ *be the last ("largest").*

### Example

*Start:*

$$\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3} \Rightarrow \overleftarrow{1}\ \overleftarrow{3}\ \overleftarrow{2} \Rightarrow \overleftarrow{3}\ \overleftarrow{1}\ \overleftarrow{2} \Rightarrow \overrightarrow{3}\ \overleftarrow{2}\ \overleftarrow{1} \Rightarrow \overleftarrow{2}\ \overrightarrow{3}\ \overleftarrow{1} \Rightarrow \overleftarrow{2}\ \overleftarrow{1}\ \overrightarrow{3}$$

### Remark (Generating Combinations)

*We wish to generate all $r$-combinations of an $n$-set $\{a_1, a_2, \ldots, a_n\}$. We shall proceed lexicographically: $\{a_1, a_2, \ldots a_r\}$ will be the first ("smallest") and $\{a_{n-r+1}, \ldots, a_n\}$ be the last ("largest").*

### Question

*What is the 4-subset of $\{1, 2, \ldots, 8\}$ following $\{3, 5, 7, 8\}$?*

### Example

*Start:*

$$\overleftarrow{1}\;\overleftarrow{2}\;\overleftarrow{3} \Rightarrow \overleftarrow{1}\;\overleftarrow{3}\;\overleftarrow{2} \Rightarrow \overleftarrow{3}\;\overleftarrow{1}\;\overleftarrow{2} \Rightarrow \overrightarrow{3}\;\overleftarrow{2}\;\overleftarrow{1} \Rightarrow \overleftarrow{2}\;\overrightarrow{3}\;\overleftarrow{1} \Rightarrow \overleftarrow{2}\;\overleftarrow{1}\;\overrightarrow{3}$$

### Remark (Generating Combinations)

*We wish to generate all $r$-combinations of an $n$-set $\{a_1, a_2, \ldots, a_n\}$. We shall proceed lexicographically: $\{a_1, a_2, \ldots a_r\}$ will be the first ("smallest") and $\{a_{n-r+1}, \ldots, a_n\}$ be the last ("largest").*

### Question

*What is the 4-subset of $\{1, 2, \ldots, 8\}$ following $\{3, 5, 7, 8\}$?*

### Example

*Start:*

$$\overleftarrow{1}\ \overleftarrow{2}\ \overleftarrow{3} \Rightarrow \overleftarrow{1}\ \overleftarrow{3}\ \overleftarrow{2} \Rightarrow \overleftarrow{3}\ \overleftarrow{1}\ \overleftarrow{2} \Rightarrow \overrightarrow{3}\ \overleftarrow{2}\ \overleftarrow{1} \Rightarrow \overleftarrow{2}\ \overrightarrow{3}\ \overleftarrow{1} \Rightarrow \overleftarrow{2}\ \overleftarrow{1}\ \overrightarrow{3}$$

### Remark (Generating Combinations)

*We wish to generate all $r$-combinations of an $n$-set $\{a_1, a_2, \ldots, a_n\}$. We shall proceed lexicographically: $\{a_1, a_2, \ldots a_r\}$ will be the first ("smallest") and $\{a_{n-r+1}, \ldots, a_n\}$ be the last ("largest").*

### Question

*What is the 4-subset of $\{1, 2, \ldots, 8\}$ following $\{3, 5, 7, 8\}$?*

Ans: $\{3, 6, 7, 8\}$.

# Generating Combinations

To simplify the notation, we shall assume that our universal set is $\{1, 2, \ldots, n\}$ and the numbers in the $r$ subsets are sorted.

1. Given an $r$-subset $\{a_1, a_2, \ldots, a_r\}$ locate the last index $i$ such that $a_i \neq n - r + i$.

# Generating Combinations

To simplify the notation, we shall assume that our universal set is $\{1, 2, \ldots, n\}$ and the numbers in the $r$ subsets are sorted.

1. Given an $r$-subset $\{a_1, a_2, \ldots, a_r\}$ locate the last index $i$ such that $a_i \neq n - r + i$.
2. Replace $a_i$ with $a_i + 1$ and add the next consecutive integers to form the next $r$-subset.

# Generating Combinations

To simplify the notation, we shall assume that our universal set is $\{1, 2, \ldots, n\}$ and the numbers in the $r$ subsets are sorted.

1. Given an $r$-subset $\{a_1, a_2, \ldots, a_r\}$ locate the last index $i$ such that $a_i \neq n - r + i$.
2. Replace $a_i$ with $a_i + 1$ and add the next consecutive integers to form the next $r$-subset.

**Example**

*The 4-combination following the combination* $\{3, 5, 7, 10\}$ *in* $\binom{\{1,2,\ldots,10\}}{4}$ *is:* $\{3, 5, 8, 9\}$.

# The Binomial theorem

You probably know a few proofs of the classical binoial theorem:

**Theorem**

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

# The Binomial theorem

You probably know a few proofs of the classical binoial theorem:

**Theorem**

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

# The Binomial theorem

You probably know a few proofs of the classical binoial theorem:

**Theorem**

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

$\binom{n}{k}$ are the binomial coefficients. A simple counting argument shows that the number of ways to select a set of $k$ objects from a set of $n$ objects is $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

# The Binomial theorem

You probably know a few proofs of the classical binoial theorem:

**Theorem**

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

$\binom{n}{k}$ are the binomial coefficients. A simple counting argument shows that the number of ways to select a set of $k$ objects from a set of $n$ objects is $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

There are many interesting relations among the binomial coefficieints. We shall briefly explore them and also see the technique of *double counting* used to prove many combinatorial identities.

# The Binomial theorem

You probably know a few proofs of the classical binoial theorem:

**Theorem**

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

$\binom{n}{k}$ are the binomial coefficients. A simple counting argument shows that the number of ways to select a set of $k$ objects from a set of $n$ objects is $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

There are many interesting relations among the binomial coefficieints. We shall briefly explore them and also see the technique of *double counting* used to prove many combinatorial identities. We start with **Pascal's idenitity:**

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

Proof.

Here is a simple combinatorial (double counting) proof:

□

**Proof.**

Here is a simple combinatorial (double counting) proof:

1. $\binom{n+1}{k}$ is the number of ways to select $k$ object from a set of $n+1$ objects.

□

Proof.

Here is a simple combinatorial (double counting) proof:

1. $\binom{n+1}{k}$ is the number of ways to select $k$ object from a set of $n+1$ objects.
2. $\binom{n}{k-1}$ is the number of ways to select $k$ objects such that each selection includes object number $n+1$.

□

**Proof.**

Here is a simple combinatorial (double counting) proof:

1. $\binom{n+1}{k}$ is the number of ways to select $k$ object from a set of $n+1$ objects.
2. $\binom{n}{k-1}$ is the number of ways to select $k$ objects such that each selection includes object number $n+1$.
3. $\binom{n}{k}$ is the number of ways to choose $k$ object that do not include object number $n+1$.

□

Proof.

Here is a simple combinatorial (double counting) proof:

1. $\binom{n+1}{k}$ is the number of ways to select $k$ object from a set of $n+1$ objects.
2. $\binom{n}{k-1}$ is the number of ways to select $k$ objects such that each selection includes object number $n+1$.
3. $\binom{n}{k}$ is the number of ways to choose $k$ object that do not include object number $n+1$.

□

Proof.

Here is a simple combinatorial (double counting) proof:

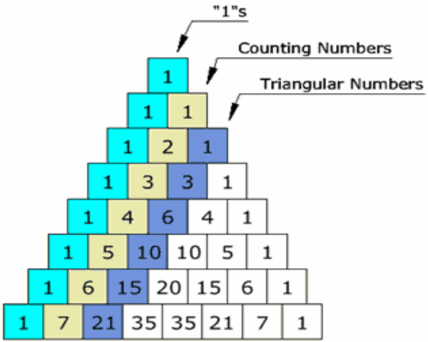
1. $\binom{n+1}{k}$ is the number of ways to select $k$ object from a set of $n+1$ objects.
2. $\binom{n}{k-1}$ is the number of ways to select $k$ objects such that each selection includes object number $n+1$.
3. $\binom{n}{k}$ is the number of ways to choose $k$ object that do not include object number $n+1$.

□

This relation among the binomial coefficient is traditionally encapsulated in the famous Pascal's triangle.

# Pascal's Triangle

Pascal's Triangle contains many patterns and relations.



# A Sample of Combinatorial Identies

There are literally thousands of combinatorial identities based on the binomial coefficients. We shall look at a small sample.

1. $$\sum_{i=0}^{n} \binom{n}{i} = 2^n$$

   (or the number of distinct subsets of an $n$-set is $2^n$).

# A Sample of Combinatorial Identies

There are literally thousands of combinatorial identities based on the binomial coefficients. We shall look at a small sample.

1. $$\sum_{i=0}^{n} \binom{n}{i} = 2^n$$

   (or the number of distinct subsets of an $n$-set is $2^n$).

2. $$\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{2i} = \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{2i-1}$$

   (or the number of ditinct subsets of even order is equal to the number of subset of odd order). Proof: $(1-1)^n = 0$.

1. $$\binom{2n}{n} = \sum_{i=0}^{n} \binom{n}{i}^2$$

1. $$\binom{2n}{n} = \sum_{i=0}^{n} \binom{n}{i}^2$$

1.
$$\binom{2n}{n} = \sum_{i=0}^{n} \binom{n}{i}^2$$

Proof:

$$\binom{2n}{n} = \sum_{i=0}^{n} \binom{n}{i} \binom{n}{n-i}$$

Both sides count the number of ways to select a team of $n$ students from a class with $n$ male students and $n$ females.

2. Vandermonde's Identity:

$$\binom{n+m}{r} = \sum_{k=0}^{r} \binom{n}{k} \binom{m}{r-k}$$

1. $$\binom{2n}{n} = \sum_{i=0}^{n} \binom{n}{i}^2$$

   Proof:

   $$\binom{2n}{n} = \sum_{i=0}^{n} \binom{n}{i}\binom{n}{n-i}$$

   Both sides count the number of ways to select a team of $n$ students from a class with $n$ male students and $n$ females.

2. Vandermonde's Identity:

   $$\binom{n+m}{r} = \sum_{k=0}^{r} \binom{n}{k}\binom{m}{r-k}$$

3. $$\sum_{k=0}^{r} \binom{n+k}{r} = \binom{n+r+1}{r}$$

# A tribute to Gauss

**Question**

*An urn contains* $100$ *balls numbered* $1, 2, \ldots, 100$*.* $100$ *persons draw a ball, note the number on it and return it to the urn. What is the probability that no two persons draw the same ball?*

# A tribute to Gauss

## Question

*An urn contains* $100$ *balls numbered* $1, 2, \ldots, 100$. $100$ *persons draw a ball, note the number on it and return it to the urn. What is the probability that no two persons draw the same ball?*

## Answer

*There are* $100^{100}$ *different ways to draw* $100$ *balls. There are only* $100!$ *ways to draw different balls. So the probability that no two persons will draw the same ball is* $\frac{100!}{100^{100}}$. *So we need to estimate this number.*

# Estimates

1. Simplest estimates:

$$n! = \prod_{i=1}^{n} i \leq \prod_{i=1}^{n} n = n^n \qquad n! = \prod_{i=1}^{n} i \geq \prod_{i=1}^{n} 2 = 2^n$$

# Estimates

1. Simplest estimates:

$$n! = \prod_{i=1}^{n} i \leq \prod_{i=1}^{n} n = n^n \qquad n! = \prod_{i=1}^{n} i \geq \prod_{i=1}^{n} 2 = 2^n$$

2. Slightly better estimates:

$$n! \geq \prod_{i=n/2}^{n} i \geq \prod_{i=n/2}^{n} n/2 = \left(\frac{n}{2}\right)^{\frac{n}{2}} \quad n! \leq \left(\prod_{i=1}^{n/2} \frac{n}{2}\right)\left(\prod_{i=n/2}^{n} n\right) = \frac{n^n}{2^{\frac{n}{2}}}$$

# Estimates

1. Simplest estimates:

$$n! = \prod_{i=1}^{n} i \leq \prod_{i=1}^{n} n = n^n \qquad n! = \prod_{i=1}^{n} i \geq \prod_{i=1}^{n} 2 = 2^n$$

2. Slightly better estimates:

$$n! \geq \prod_{i=n/2}^{n} i \geq \prod_{i=n/2}^{n} n/2 = \left(\frac{n}{2}\right)^{\frac{n}{2}} \quad n! \leq \left(\prod_{i=1}^{n/2} \frac{n}{2}\right)\left(\prod_{i=n/2}^{n} n\right) = \frac{n^n}{2^{\frac{n}{2}}}$$

## Remark

*So the probability that each person will see a different number is* $< 2^{-50}$ *or just about no chance!*

*Even though it looks as if the estimates assume that n is even, it is not difficult to show that they hold for odd n.*

# Gauss' nice estimates

Theorem (Gauss)

$$n^{\frac{n}{2}} \leq n! \leq \left(\frac{n+1}{2}\right)^n$$

# Gauss' nice estimates

Theorem (Gauss)

$$n^{\frac{n}{2}} \leq n! \leq \left(\frac{n+1}{2}\right)^n$$

Proof.

# Gauss' nice estimates

Theorem (Gauss)

$$n^{\frac{n}{2}} \leq n! \leq \left(\frac{n+1}{2}\right)^n$$

Proof.

$$\prod_{i=1}^{n} i = \prod_{i=1}^{n}(n+1-i) = n! \Rightarrow n! = \sqrt{\prod_{i=1}^{n} i(n+1-i)}$$

# Gauss' nice estimates

**Theorem (Gauss)**

$$n^{\frac{n}{2}} \leq n! \leq \left(\frac{n+1}{2}\right)^n$$

**Proof.**

$$\prod_{i=1}^{n} i = \prod_{i=1}^{n}(n+1-i) = n! \Rightarrow n! = \sqrt{\prod_{i=1}^{n} i(n+1-i)}$$

By the geometric-arithmetic inequality $\sqrt{i(n+1-i)} \leq \left(\frac{n+1}{2}\right)$ so

$$n! \leq \prod_{i=1}^{n} \frac{n+1}{2} = \left(\frac{n+1}{2}\right)^n$$

□

# Gauss' nice estimates

**Theorem (Gauss)**

$$n^{\frac{n}{2}} \leq n! \leq \left(\frac{n+1}{2}\right)^n$$

**Proof.**

$$\prod_{i=1}^{n} i = \prod_{i=1}^{n}(n+1-i) = n! \Rightarrow n! = \sqrt{\prod_{i=1}^{n} i(n+1-i)}$$

By the geometric-arithmetic inequality $\sqrt{i(n+1-i)} \leq \left(\frac{n+1}{2}\right)$ so

$$n! \leq \prod_{i=1}^{n} \frac{n+1}{2} = \left(\frac{n+1}{2}\right)^n$$

$i(n+1-i) \geq n \Rightarrow n! \geq \sqrt{n^n}$

We conclude by mentioning a very famous and beautiful approximation: Stirling's Formula.
It uses two of the most famous constants in mathematics: $\pi$ and $e$ in one expression involving an approximation of the integer valued function $n!$.

$$n! \sim \sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n}$$

For a proof of this formula see the file Stirling.pdf.

We conclude by mentioning a very famous and beautiful approximation: Stirling's Formula.
It uses two of the most famous constants in mathematics: $\pi$ and $e$ in one expression involving an approximation of the integer valued function $n!$.

$$n! \sim \sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n}$$

For a proof of this formula see the file Stirling.pdf.
In many applications, for example in combinatorial probability, factorials and binomial coefficients are very ubiquitous. Stirling's formula provides an excellent approximation using a much easier expression to manipulate. For instance, how big is 100!?

We conclude by mentioning a very famous and beautiful approximation: Stirling's Formula.
It uses two of the most famous constants in mathematics: $\pi$ and $e$ in one expression involving an approximation of the integer valued function $n!$.

$$n! \sim \sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n}$$

For a proof of this formula see the file Stirling.pdf.
In many applications, for example in combinatorial probability, factorials and binomial coefficients are very ubiquitous. Stirling's formula provides an excellent approximation using a much easier expression to manipulate. For instance, how big is 100!?

Using Stirling's formula we get:

$$\lg 100! \approx 100 \lg(\frac{100}{e}) + 1 + \lg \sqrt{2\pi} = 157.96 \ldots$$

The actual number of digits of 100! is 158.

We conclude by mentioning a very famous and beautiful approximation: Stirling's Formula.
It uses two of the most famous constants in mathematics: $\pi$ and $e$ in one expression involving an approximation of the integer valued function $n!$.

$$n! \sim \sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n}$$

For a proof of this formula see the file Stirling.pdf.
In many applications, for example in combinatorial probability, factorials and binomial coefficients are very ubiquitous. Stirling's formula provides an excellent approximation using a much easier expression to manipulate. For instance, how big is 100!?

Using Stirling's formula we get:

$$\lg 100! \approx 100 \lg(\frac{100}{e}) + 1 + \lg \sqrt{2\pi} = 157.96 \ldots$$

The actual number of digits of 100! is 158.

$\binom{2n}{n} \sim \frac{4^n}{\sqrt{2\pi}}$ Is another useful approximation.