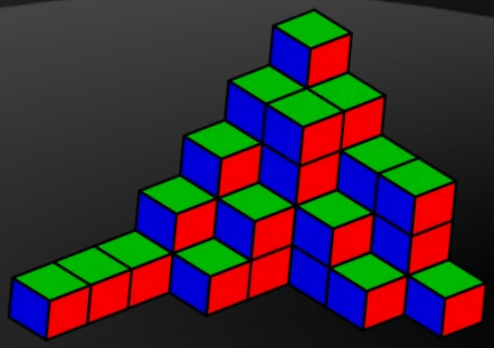
Telerik Academy



Combinatorics

Brief Overview of Combinations, Permutations and Binary Vectors

Data Structures and Algorithms

Telerik Software Academy

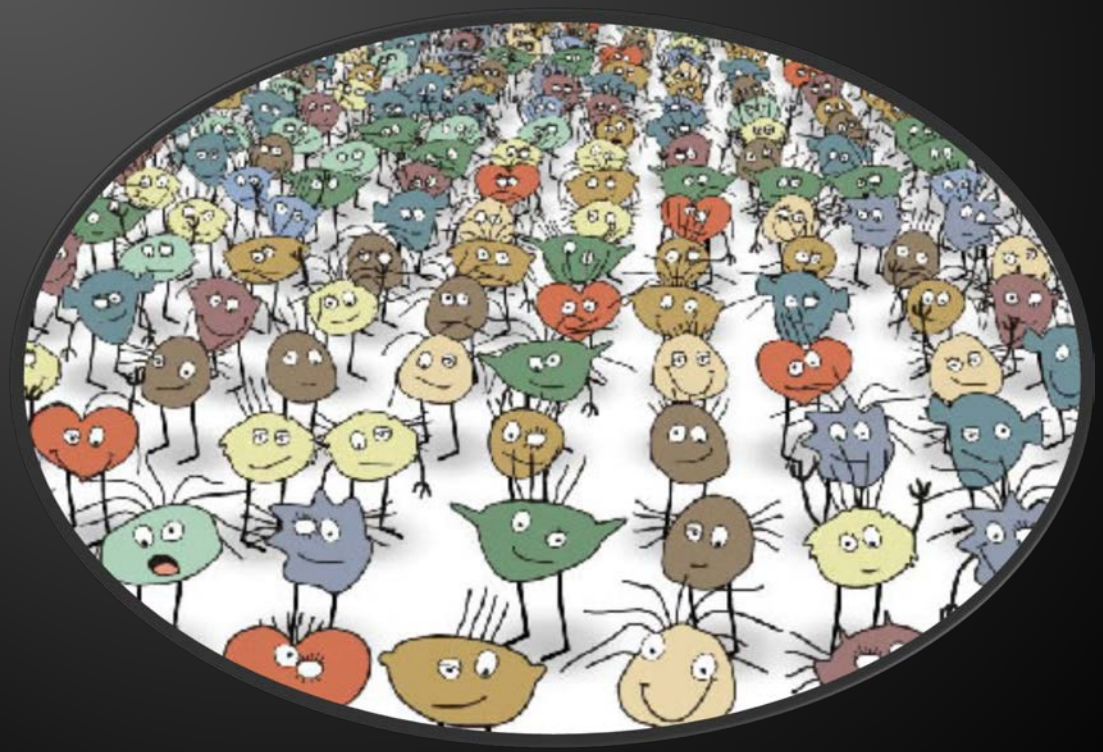
http://academy.telerik.com

Table of Contents

1. Definitions in Combinatorics
2. Permutations
3. Combinations
   - Pascal's Triangle
4. Binary Vectors
5. Resources

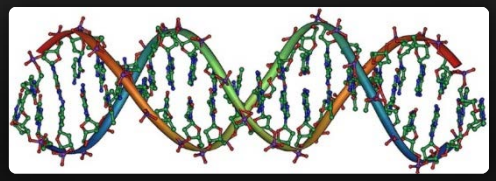
Telerik Academy

Definitions in Combinatorics



Combinatorics

- Combinatorics is a branch of mathematics
  - Concerning the study of finite or countable discrete structures
- Combinatorial problems arise in many areas of pure mathematics
  - Notably in algebra, probability theory, topology, geometry, physics, chemistry, biology, etc.



Combinations

- *"My fruit salad is a combination of grapes, strawberries and bananas"*
  - We don't care what order the fruits are in
    - "*bananas, grapes and strawberries*" or "*grapes, bananas and strawberries*" → it is the same salad

- If the order doesn't matter, it is a combination

Permutations / Variations

- *"The combination to the safe is 4385"*.
  - Now we do care about the order
  - "*8453*" would not work, nor would "*4538*"
  - It has to be exactly *4-3-8-5*

- If the order does matter it is a permutation / variation
  - A permutation / variation is an ordered Combination
- Easy to remember: "Permutation … Position"

Factorial

- The factorial function (!) just means to multiply a series of descending natural numbers
  - n! = n × (n-1)!
    - 1! = 1, 0! = 1
    - 4! = 4 × 3 × 2 × 1 = 24
    - 7! = 7 × 6 × 5 × 4 × 3 × 2 × 1 = 5040

$$n! = \prod_{k=1}^{n} k$$

$$n! = \begin{cases} 1 & \text{if } n = 0, \\ (n-1)! \times n & \text{if } n > 0. \end{cases}$$

Factorial - Source code

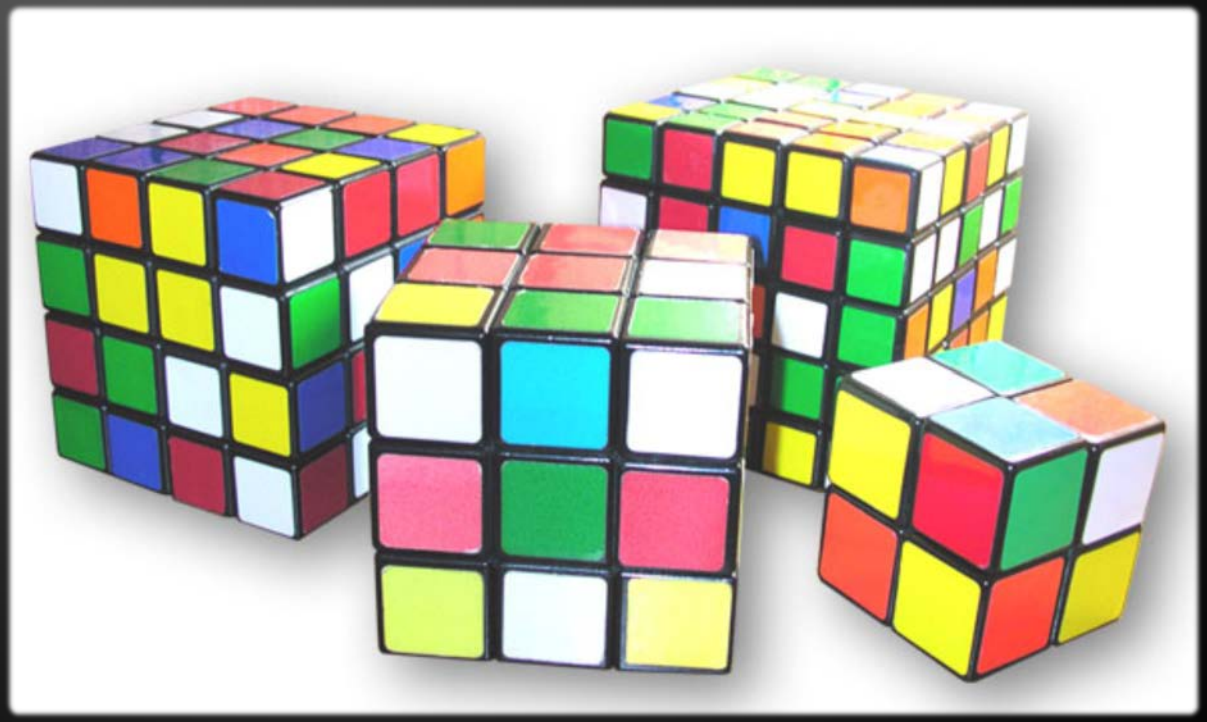
Factorial – recursive

```
static long Factorial(int n)
{
    if (n == 0) return 1;
    return Factorial(n - 1) * n;
}
```

Factorial – iterative

```
static long Factorial(int n)
{
    long result = 1;
    for(int i = 2; i <= n; i++)
        result = result * i;
    return result;
}
```

Variations



Variations

- Variations (with repetitions)
  - Easiest to calculate
- When you have *n* things to choose from … you have *n* choices each time!
- When choosing *k* of them, the variations are:
  - $n \times n \times \ldots \times n$ (*k times*)

$$n^k$$

Apple	Banana	Orange
Apple	Apple	Apple
Apple	Apple	Banana
Apple	Apple	Orange
Apple	Banana	Apple
Apple	Banana	Banana
Apple	Banana	Orange
Apple	Orange	Apple
Apple	Orange	Banana
Apple	Orange	Orange
Banana	Apple	Apple
Banana	Apple	Banana
Banana	Apple	Orange
Banana	Banana	Apple
Banana	Banana	Banana
Banana	Banana	Orange
Banana	Orange	Apple
Banana	Orange	Banana
Banana	Orange	Orange
Orange	Apple	Apple
Orange	Apple	Banana
Orange	Apple	Orange
Orange	Banana	Apple
Orange	Banana	Banana
Orange	Banana	Orange
Orange	Orange	Apple
Orange	Orange	Banana
Orange	Orange	Orange

Variations (2)

- Example: in the lock below, there are 10 numbers to choose from (0, 1, … 9) and you choose 3 of them:
  - 10 × 10 × 10 (3 times) = 10^3 = 1 000 variations

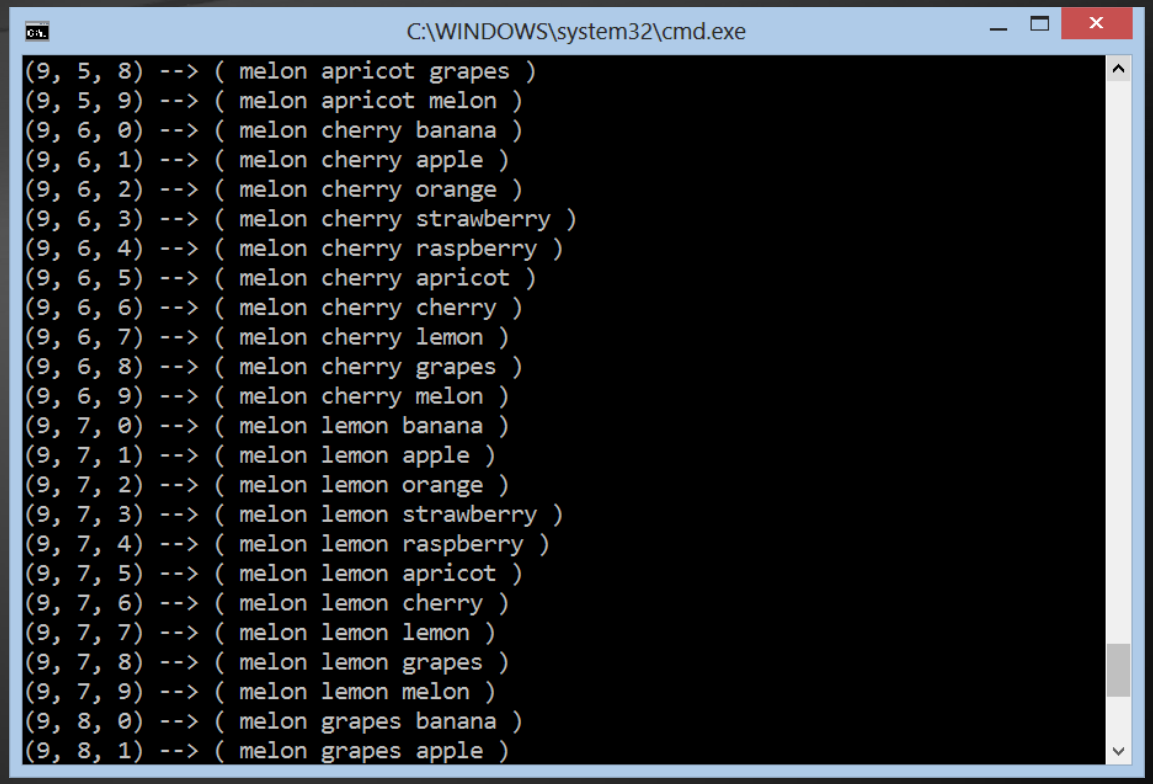
- All variations from (0, 0, 0) to (9, 9, 9)

Generating Variations

```
static void Main()
{
  GenerateVariations(0);
}

static void GenerateVariations(int index)
{
  if (index >= k)
    Print(arr);
  else
    for (int i = 0; i < n; i++)
    {
      arr[index] = i;
      GenerateVariations(index + 1);
    }
}
```

```
1 1 1
1 1 2
1 1 3
1 2 1
1 2 2
1 2 3
1 3 1
1 3 2
1 3 3
2 1 1
2 1 2
2 1 3
2 2 1
2 2 2
2 2 3
2 3 1
2 3 2
2 3 3
3 1 1
3 1 2
3 1 3
3 2 1
3 2 2
3 2 3
3 3 1
3 3 2
3 3 3
```

Telerik Academy



Variations with Repetitions

Live Demo

Variations without Repetition

- Suppose we have 16 billiard balls
  - But maybe you don't want to choose them all, just 3 of them, so that would be only
    - 16 × 15 × 14 = 3360
    - There are 3360 different ways that 3 pool balls could be selected out of 16 balls
  - 16! / 13! = 16 × 15 × 14

$$\frac{n!}{(n-k)!}$$

where *n* is the number of things to choose from, and you choose *k* of them
(No repetition, order matters)

Telerik Academy

Variations without Repetition

- Example:
  - How many words of 2 different letters can you make with 4 letters { a, b, c, d }?

```
ab, ac, ad, ba, bc, bd, ca, cb, cd, da, db, dc
```

- How to generate variations without repetitions?

$V_2^4 = 12$

$$\begin{bmatrix} - & ab & ac & ad \\ ba & - & bc & bd \\ ca & cb & - & cd \\ da & db & dc & - \end{bmatrix}$$

  - The same way like variations with repetitions
  - Just use each element at most once

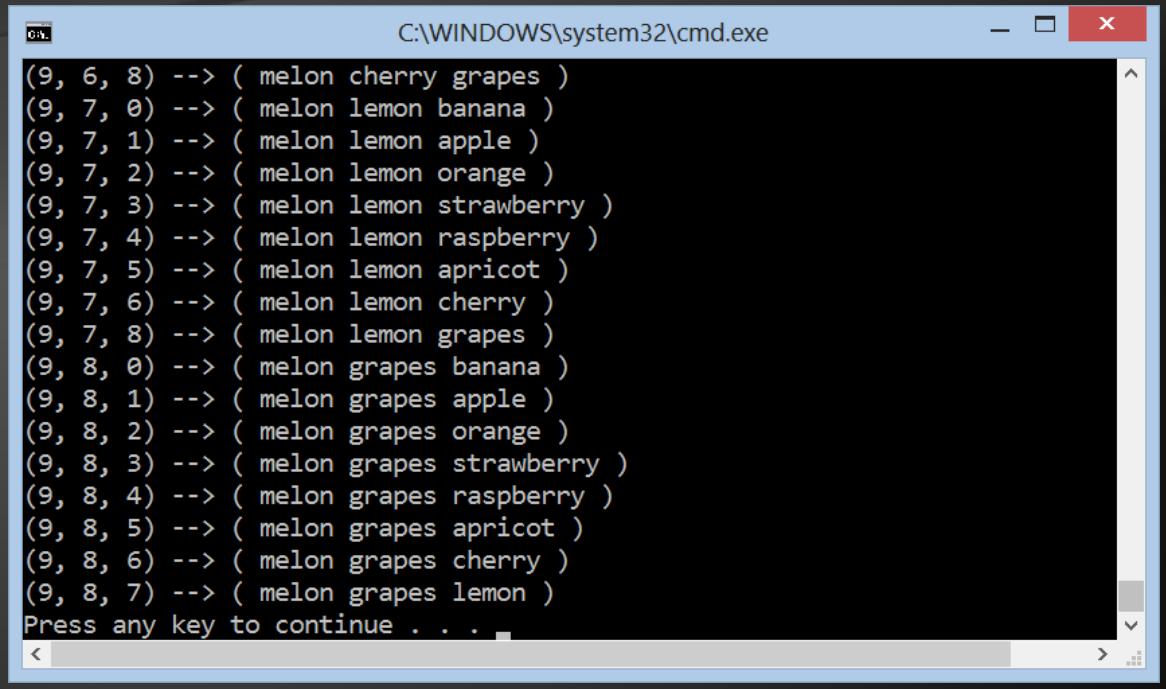
Telerik Academy

Generating Variations without Repetitions

```
static void GenerateVariationsNoReps(int index)
{
  if (index >= k)
    PrintVariations();
  else
    for (int i = 0; i < n; i++)
      if (!used[i])
      {
        used[i] = true;
        arr[index] = i;
        GenerateVariationsNoReps(index + 1);
        used[i] = false;
      }
}

GenerateVariationsNoReps(0);
```

```
(0, 1)
(0, 2)
(0, 3)
(1, 0)
(1, 2)
(1, 3)
(2, 0)
(2, 1)
(2, 3)
(3, 0)
(3, 1)
(3, 2)
```

Telerik Academy

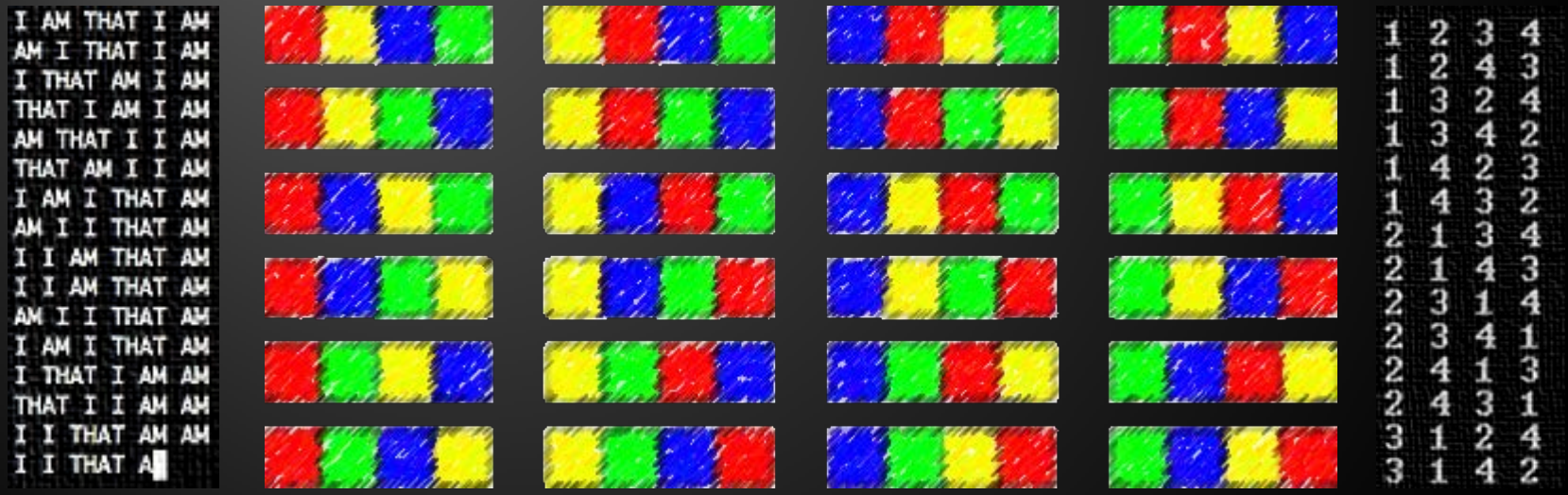


Variations without Repetitions

Live Demo

Telerik Academy

Permutations



Permutations

- Less number of available choices each time
- What order could 16 pool balls be in?

- After choosing, ball 9 we can't choose the same ball again
  - First choice = 16 possibilities
  - Second choice = 15 possibilities, etc., etc.
- Total permutations:
  - $16 \times 15 \times 14 \times \ldots \times 2 \times 1 = 16! = 20\,922\,789\,888\,000$

Generating Permutations

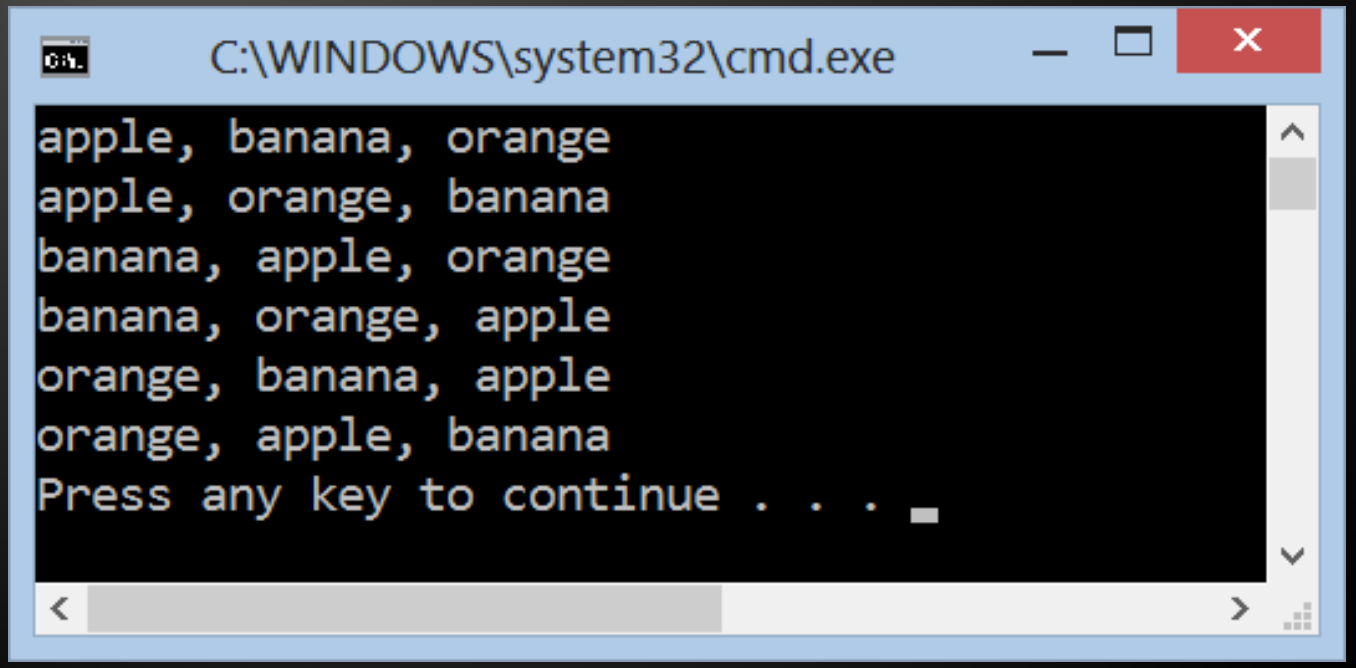
```
static void Perm<T>(T[] arr, int k)
{
  if (k >= arr.Length)
    Print(arr);
  else
  {
    Perm(arr, k + 1);
    for (int i = k + 1; i < arr.Length; i++)
    {
      Swap(ref arr[k], ref arr[i]);
      Perm(arr, k + 1);
      Swap(ref arr[k], ref arr[i]);
    }
  }
}
```

```
1 2 3 4
1 2 4 3
1 3 2 4
1 3 4 2
1 4 2 3
1 4 3 2
2 1 3 4
2 1 4 3
2 3 1 4
2 3 4 1
2 4 1 3
2 4 3 1
3 1 2 4
3 1 4 2
3 2 1 4
3 2 4 1
3 4 1 2
3 4 2 1
4 1 2 3
4 1 3 2
4 2 1 3
4 2 3 1
4 3 1 2
4 3 2 1
```

Telerik Academy

Generating Permutations

Live Demo



Permutations with Repetitions

- We have a set of elements, with repetitions
  - E. g. set = { 3, 5, 1, 5, 5 }
- We want to generate all unique permutations (without duplicates):

{ 1, 3, 5, 5, 5 } { 1, 5, 3, 5, 5 } { 1, 5, 5, 3, 5 } { 1, 5, 5, 5, 3 }
{ 3, 1, 5, 5, 5 } { 3, 5, 1, 5, 5 } { 3, 5, 5, 1, 5 } { 3, 5, 5, 5, 1 }
{ 5, 1, 3, 5, 5 } { 5, 1, 5, 3, 5 } { 5, 1, 5, 5, 3 } { 5, 3, 1, 5, 5 }
{ 5, 3, 5, 1, 5 } { 5, 3, 5, 5, 1 } { 5, 5, 1, 3, 5 } { 5, 5, 1, 5, 3 }
{ 5, 5, 3, 1, 5 } { 5, 5, 3, 5, 1 } { 5, 5, 5, 1, 3 } { 5, 5, 5, 3, 1 }

- We want to efficiently avoid the repeating ones, i.e. to work fast for { 1, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5 }

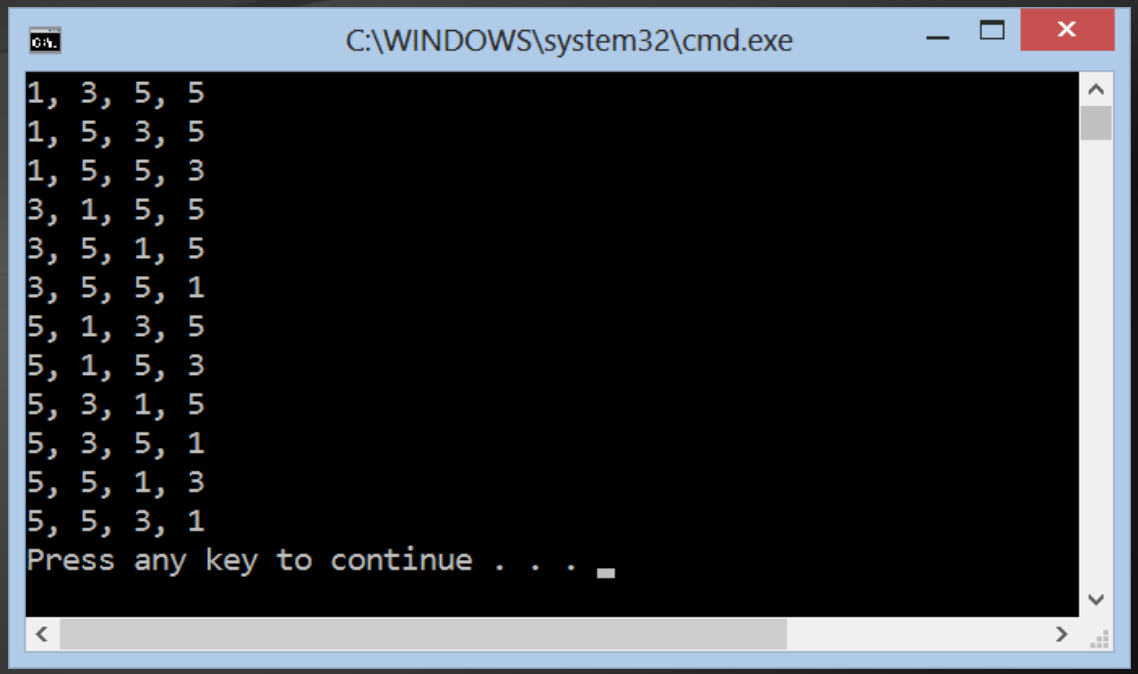
Telerik Academy

Generating Permutations with Repetitions

```
var arr = new int[] { 3, 5, 1, 5, 5 };
Array.Sort(arr);
PermuteRep(arr, 0, arr.Length);

static void PermuteRep(int[] arr, int start, int n)
{
  Print(arr);

  for (int left = n - 2; left >= start; left--)
  {
    for (int right = left + 1; right < n; right++)
      if (arr[left] != arr[right])
      {
        Swap(ref arr[left], ref arr[right]);
        PermuteRep(arr, left + 1, n);
      }

    var firstElement = arr[left];
    for (int i = left; i < n - 1; i++)
      arr[i] = arr[i + 1];
    arr[n - 1] = firstElement;
  }
}
```

```
1, 3, 5, 5, 5
1, 5, 3, 5, 5
1, 5, 5, 3, 5
1, 5, 5, 5, 3
3, 1, 5, 5, 5
3, 5, 1, 5, 5
3, 5, 5, 1, 5
3, 5, 5, 5, 1
5, 1, 3, 5, 5
5, 1, 5, 3, 5
5, 1, 5, 5, 3
5, 3, 1, 5, 5
5, 3, 5, 1, 5
5, 3, 5, 5, 1
5, 5, 1, 3, 5
5, 5, 1, 5, 3
5, 5, 3, 1, 5
5, 5, 3, 5, 1
5, 5, 5, 1, 3
5, 5, 5, 3, 1
```

Telerik Academy

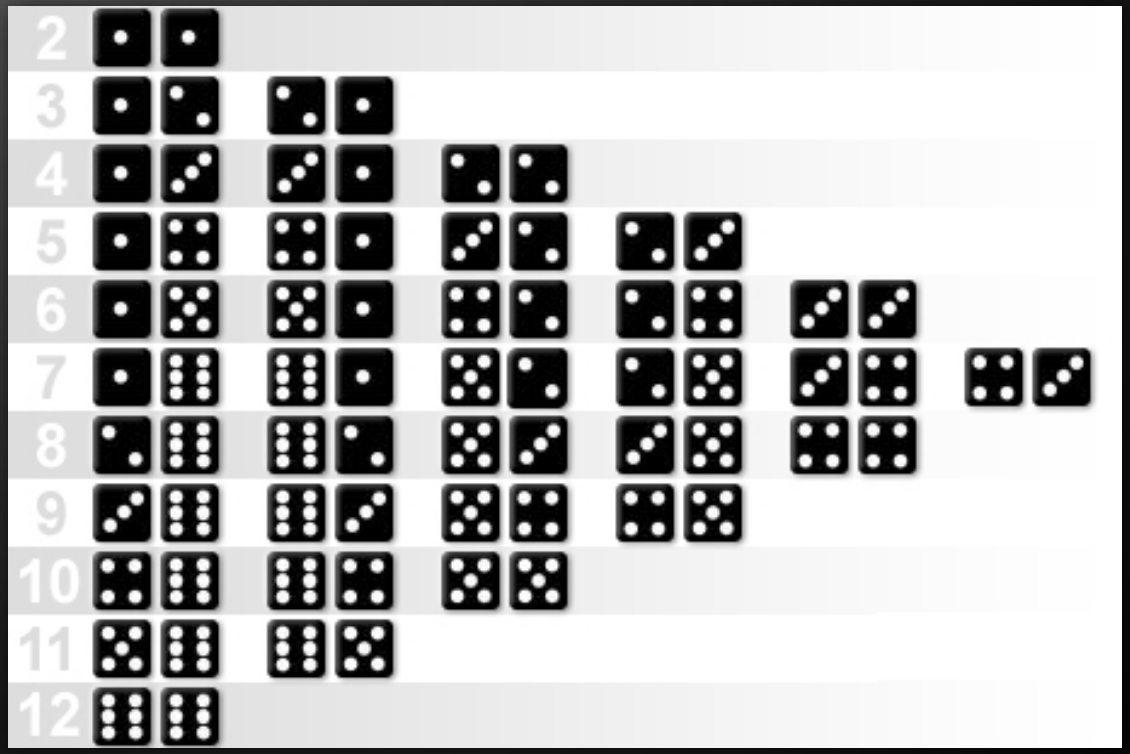


Generating Permutations with Repetitions

Live Demo

Telerik Academy

Combinations



Combinations

- Order does not matter!
- Two types of combinations:
  - Repetition is allowed
    - Coins in your pocket
    - 5,5,20,20,20,10,10
  - No repetition
    - Lottery numbers
    - TOTO 6/49, 6/42, 5/35
    - 2,14,15,27,30,33

Combinations without Repetition

- Back to the 3 of 16 billiard balls
  - Many comb. will be the same
  - We don't care about the order

Order does matter	Order doesn't matter
1 2 3	1 2 3
1 3 2	
2 1 3	
2 3 1	
3 1 2	
3 2 1	

- Permutations w/o repetitions reduced by how many ways the objects could be in order:

$$C(n,k) = \binom{n}{k} = \frac{n!}{(n-k)!\,k!}$$

- This is how lotteries work (TOTO 6/49)
- Often called "n choose k" (Google it!)

Generate Combinations without Repetitions

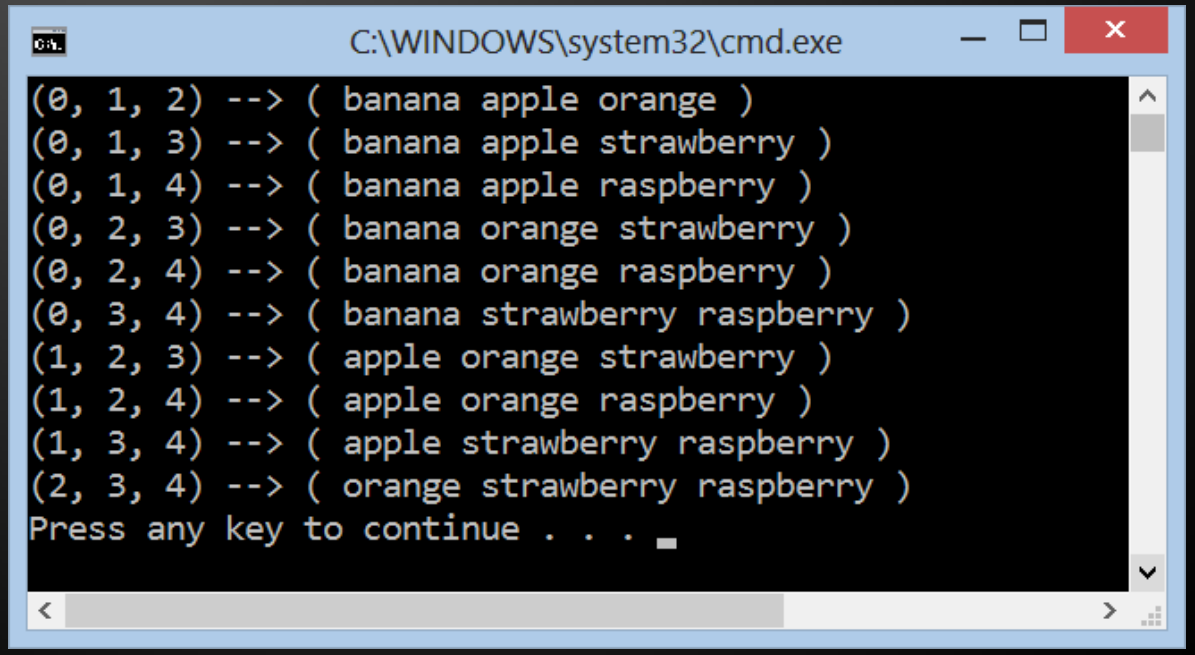
```
static void Main()
{
  Comb(0, 0);
}

static void Comb(int index, int start)
{
  if (index >= k)
    PrintCombinations();
  else
    for (int i = start; i < n; i++)
    {
      arr[index] = i;
      Comb(index + 1, i + 1);
    }
}
```

```
C(5,3):
1 2 3
1 2 4
1 2 5
1 3 4
1 3 5
1 4 5
2 3 4
2 3 5
2 4 5
3 4 5
```

Telerik Academy

Combinations without Repetitions

Live Demo

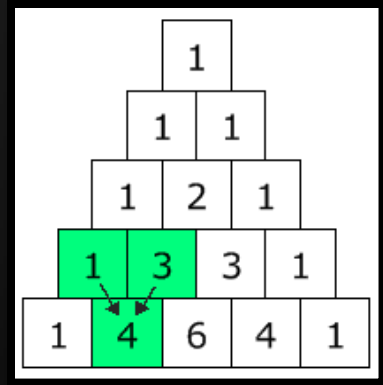


Pascal's Triangle

- The Pascal's triangle shows you how many combinations of objects without repetition are possible

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

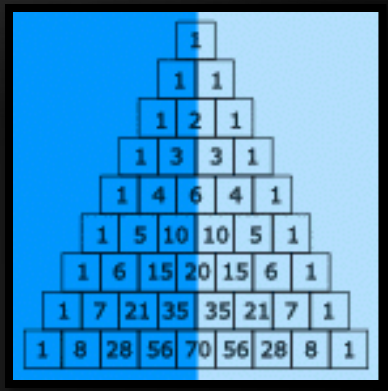
  - Go down to row "n" (the top row is 0)
  - Move along "k" places
  - The value there is your answer

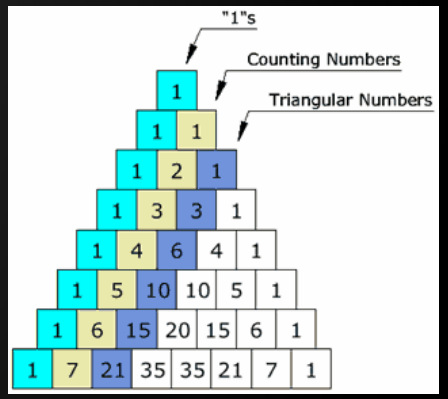


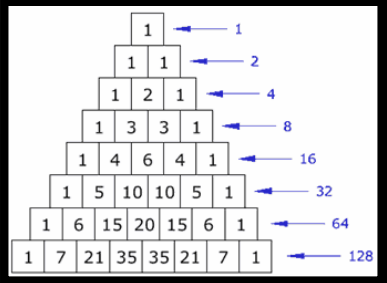
- Build the triangle: start with "1" at the top, then continue placing numbers below it in a triangular pattern

Pascal's Triangle's (2)

- The triangle is symmetrical
  - Numbers on the left side have identical matching numbers on the right side, like a mirror image
- Diagonals:
  - First diagonal – only "1"s
  - Second diagonal – 1, 2, 3, etc.
  - Third diagonal – triangular numbers
- Horizontal sums: powers of 2







Telerik Academy

Binomial Coefficients

Calculate using recursion:

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

```
static decimal Binom(int n, int k)
{  if (k > n) return 0;
   else if (0 == k || k == n) return 1;
   else return Binom(n-1, k-1) + Binom(n-1, k);
}
```

Calculate using dynamic programming:

```
decimal m[MAX], i, j;
for (i = 0; i <= n; i++) {
   m[i] = 1;
   if (i > 1) {
      if (k < i - 1) j = k; else j = i - 1;
      for (; j >= 1; j--) m[j] += m[j - 1];
   }
} // The answer is in m[k]
```

Combinations with Repetition

- Ice-cream example

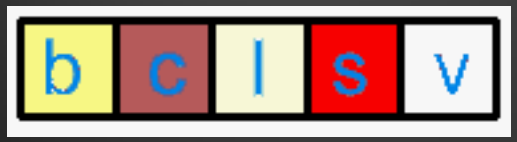
  - 5 flavors of ice-cream: banana, chocolate, lemon, strawberry and vanilla
  - 3 scoops

- How many combinations will there be?
- Let's use letters for the flavors: {b, c, l, s, v}
  - {c, c, c} (3 scoops of chocolate)
  - {b, l, v} (one each of banana, lemon and vanilla)
  - {b, v, v} (one of banana, two of vanilla)

Telerik Academy

Combinations with Repetition

- Ice-cream example
  - n=5 things to choose from, choose k=3 of them
  - Order does not matter, and we can repeat
- Think about the ice cream being in boxes



  - arrow means move, circle means scoop
  - {c, c, c} (3 scoops of chocolate)

  - {b, l, v} (banana, lemon, vanilla)

  - {b, v, v} (banana, two of vanilla)

Combinations with Repetition

- We have a *simpler* problem to solve
  - How many different ways can you arrange arrows and circles?
  - 3 circles (3 scoops) and 4 arrows (we need to move 4 times to go from the 1st to 5th container)
  - There are *k + (n-1)* positions, and we want to choose k of them to have circles

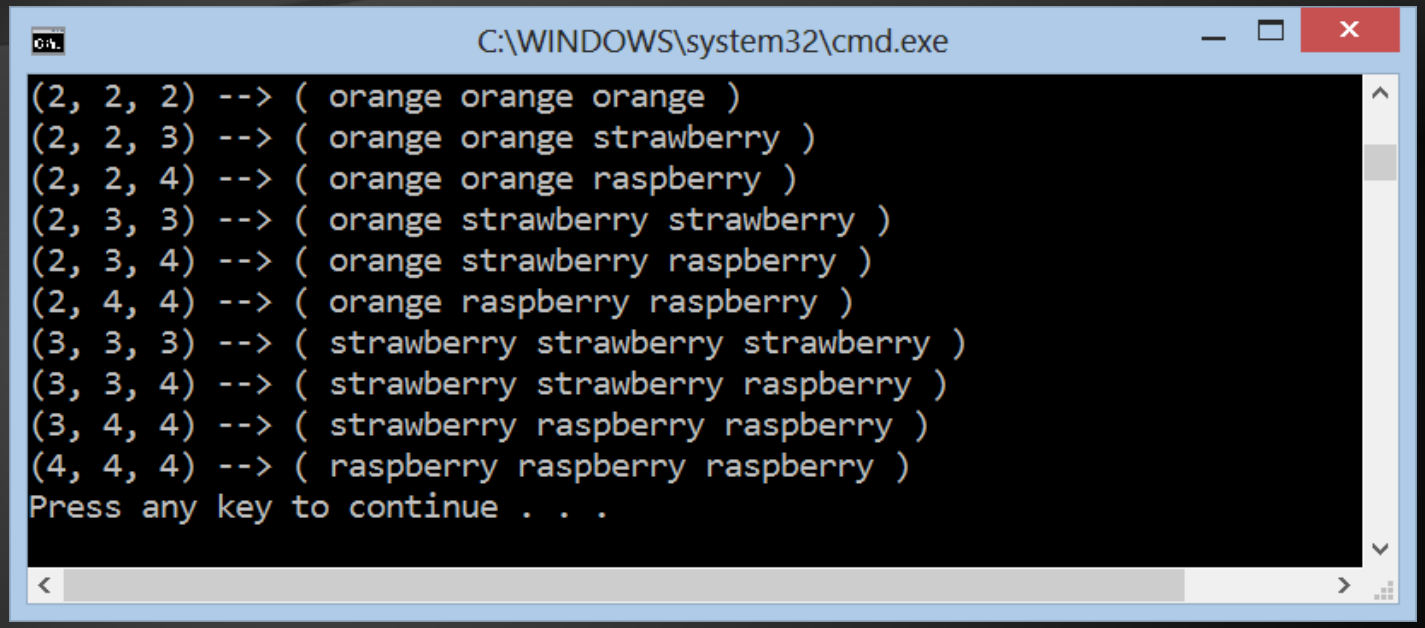
$$\binom{n+k-1}{k} = \binom{n+k-1}{n-1} = \frac{(n+k-1)!}{(n-1)!\,k!}$$

Generate Combinations with Repetitions

```
static void Main()
{
  Comb(0, 0);
}

static void CombReps(int index, int start)
{
  if (index >= k)
    PrintVariations();
  else
    for (int i = start; i < n; i++)
    {
      arr[index] = i;
      CombReps(index + 1, i);
    }
}
```

```
(0, 0)
(0, 1)
(0, 2)
(0, 3)
(0, 4)
(1, 1)
(1, 2)
(1, 3)
(1, 4)
(2, 2)
(2, 3)
(2, 4)
(3, 3)
(3, 4)
(4, 4)
```

Telerik Academy



Generating Combinations with Repetitions

Live Demo

Combinatorial Formulas

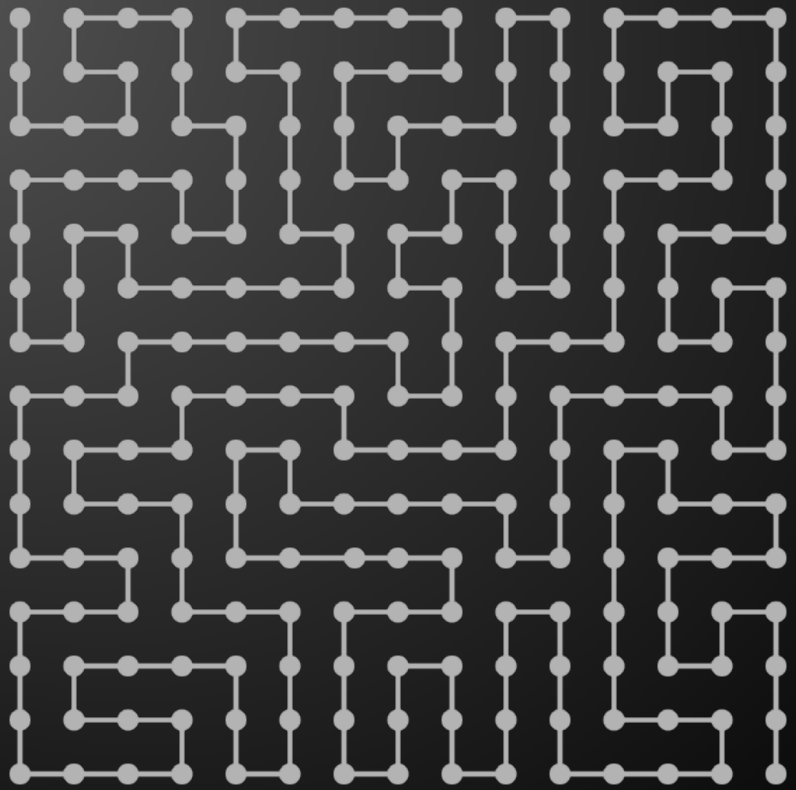
- Calculating the number of permutations, variations, combinations
  - Use the well known formulas:

	Order is important		Order is NOT important
	Arrange **Permutations**	**Arrange and Pick** **Variations**	**Pick** **Combinations**
Without repetition	$P_n = P_n^n = n!$	$V_p^n = P_p^n = \frac{n!}{(n-p)!}$	$C_p^n = \frac{n!}{p!(n-p)!}$
Example	*How many ways are there to arrange 3 letters a,b,c?* $P_3 = 3! = 6$ $\begin{bmatrix} abc & bca & cab \\ acb & bac & cba \end{bmatrix}$	*How many words of 2 different letters can you make with 4 letters a,b,c,d?* $V_2^4 = 12$ $\begin{bmatrix} - & ab & ac & ad \\ ba & - & bc & bd \\ ca & cb & - & cd \\ da & db & dc & - \end{bmatrix}$	*How many ways are there to pick 2 different letters out of 4 letters a,b,c,d? [number of subsets]* $C_2^4 = 6$ $\begin{bmatrix} - & ab & ac & ad \\ - & - & bc & bd \\ - & - & - & cd \\ - & - & - & - \end{bmatrix}$
With repetition	$P_{n_1,\ldots,n_k} = \frac{(\sum n_i)!}{\prod (n_i!)}$	$\overline{V}_p^n = n^p$	$\overline{C}_p^n = C_p^{n+p-1}$
Example	*How many ways are there to arrange 2 letters a and 2 letters b?* $P_{2,2} = \frac{(2+2)!}{2!.2!} = 6$ $\begin{bmatrix} aabb & abab & abba \\ baab & baba & bbaa \end{bmatrix}$	*How many words of 2 letters can you make with 4 letters a,b,c,d?* $\overline{V}_2^4 = 16$ $\begin{bmatrix} aa & ab & ac & ad \\ ba & bb & bc & bd \\ ca & cb & cc & cd \\ da & db & dc & dd \end{bmatrix}$	*How many ways are there to pick 2 letters out of 4 letters* $\overline{C}_2^4 = C_2^5 = 10$ $\begin{bmatrix} aa & ab & ac & ad \\ - & bb & bc & bd \\ - & - & cc & cd \\ - & - & - & dd \end{bmatrix}$ *a,b,c,d?*

Source: http://users.telenet.be/vdmoortel/dirk/Maths/PermVarComb.html

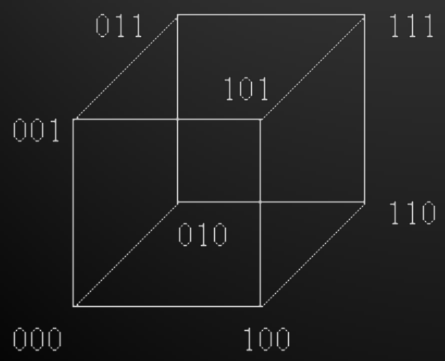
Telerik Academy

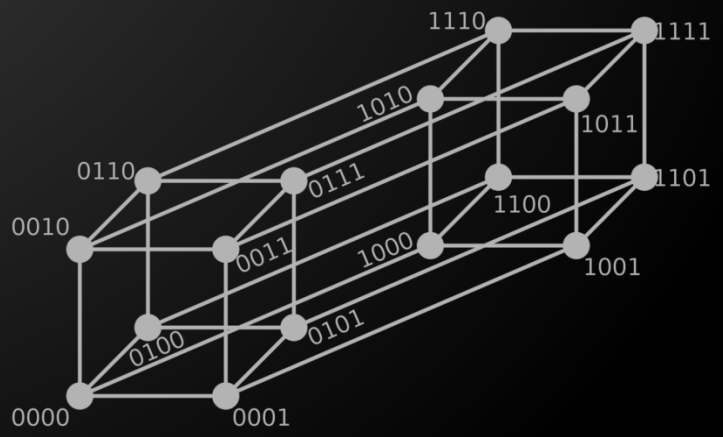
Binary Vectors



Binary Vectors

- Some problems can be reformulated in terms of binary vectors – (1, 0, 1, 1, 1, 0, …)
- Combinatorial properties of binary vectors:
  - Number of binary vectors of length n: 2^n.
  - Number of binary vectors of length n and with k `1` is C(n, k) (we choose k positions for n `1`)





Gray Code

- Gray code (a.k.a. reflected binary code) is a binary numeral system where two successive values differ by only one bit

Decimal	Gray	Binary
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

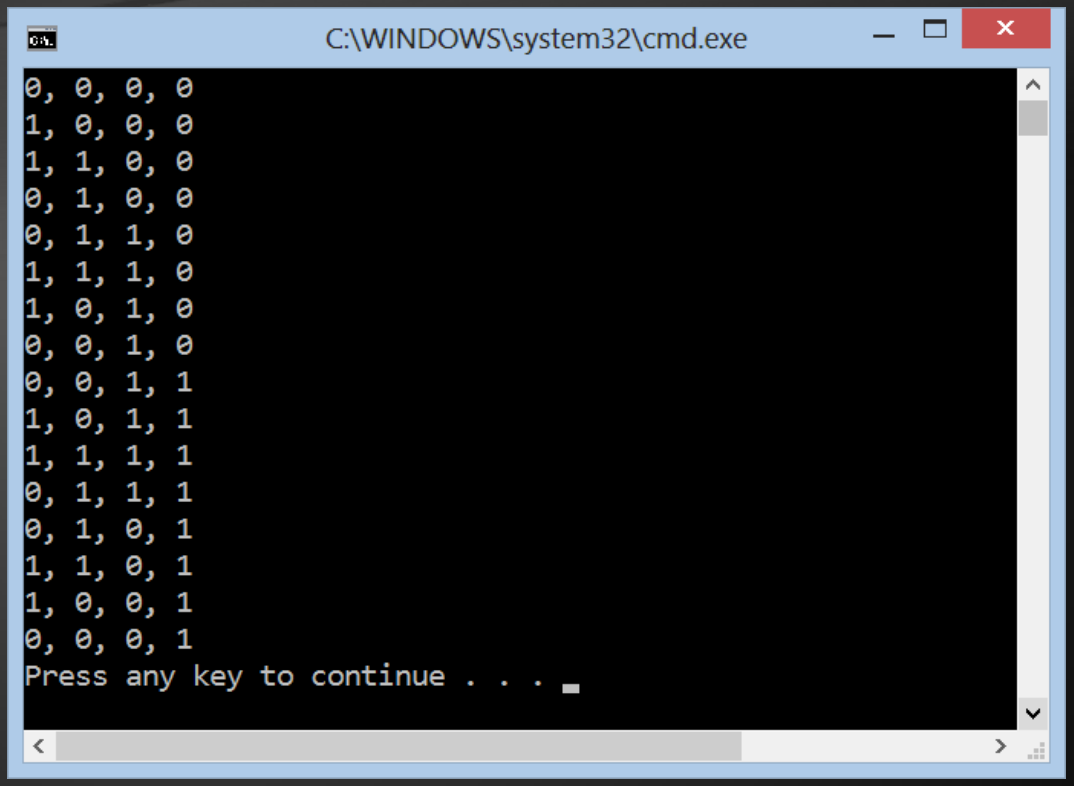
Gray code by bit width	
2-bit	4-bit
00 01 11 10	0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000
3-bit	
000 001 011 010 110 111 101 100	

Gray Code – Source Code

```
static int n = 4, a[1000], i;
static void Print()
{  for (i = n; i >= 1; i--) Console.Write("{0} ", a[i]);
   Console.WriteLine();
}
static void Backgray(int k)
{  if (0 == k) { Print(); return; }
   a[k] = 1;  Forwgray(k - 1);
   a[k] = 0;  Backgray(k - 1);
}
static void Forwgray(int k)
{  if (0 == k) { Print(); return; }
   a[k] = 0;  Forwgray(k - 1);
   a[k] = 1;  Backgray(k - 1);
}
static int Main() { Forwgray(n); return 0; }
```

```
0 0 0 0
0 0 0 1
0 0 1 1
0 0 1 0
0 1 1 0
0 1 1 1
0 1 0 1
0 1 0 0
1 1 0 0
1 1 0 1
1 1 1 1
1 1 1 0
1 0 1 0
1 0 1 1
1 0 0 1
1 0 0 0
```

Telerik Academy



Gray Code

Live Demo

Resources

- Video lectures (in Bulgarian)
  - http://cs.maycamp.com/?page_id=685
  - http://cs.maycamp.com/?page_id=760
  - http://cs.maycamp.com/?page_id=764
- TopCoder article: http://goo.gl/bN9RL
- http://en.wikipedia.org/wiki/Permutation
- http://en.wikipedia.org/wiki/Combination
- Book: http://goo.gl/Z2Knl
- Nakov's book: Programming = ++Algorithms;

Telerik Academy

Combinatorics

Questions?

http://algoacademy.telerik.com

Exercises

1. Solve 2 problems by choice from Algo Academy (October 2012 – Combinatorics):
   http://academy.telerik.com/algoacademy/season-2012-2013/training-27-28-Oct-2012-combinatorics

   You may test your solutions in BG Coder:
   http://bgcoder.com/Contest/Practice/59

Telerik Academy

Free Trainings @ Telerik Academy

- C# Programming @ Telerik Academy
  - csharpfundamentals.telerik.com

- Telerik Software Academy
  - academy.telerik.com

- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy

- Telerik Software Academy Forums
  - forums.academy.telerik.com