**Week 6**

**Combination and Permutation**

Let’s discuss first what is the difference between them and then we will explain each one of them in depth.

When we use the word combination in english we don’t think about the order of things we are talking about,

For Example:

***My fruit salad is a combination of apples, grapes and bananas"*** We don't care what order the fruits are in, they could also be "bananas, grapes and apples" or "grapes, apples and bananas", it’s the same fruit salad.

While in permutation the order differs for sure for example:

***"The combination to the safe is 472"***. Now we **do** care about the order. "724" won't work, nor will "247". It has to be exactly **4-7-2**.

So let’s take a step toward mathematical approach of combination and permutation

When the order **doesn't** matter, it is a **Combination.**  
When the order **does** matter it is a **Permutation**.

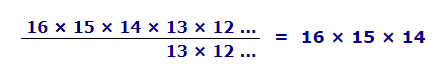


Permutation Lock

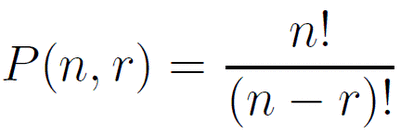
* **Permutation**There are basically two types of permutation:  
    
  - **Repetition is Allowed**: such as the lock above. It could be "333".  
  - **No Repetition**: for example the first three people in a running race. You can't be first and second.  
    
   **Permutations with Repetition:**When a thing has n different types ... we have n choices each time!  
    
  For example: choosing 3 of those things, the permutations are:  
    
  n × n × n   
  (n multiplied 3 times)  
    
  More generally: choosing r of something that has n different types, the permutations are:  
    
  n × n × ... (r times)  
    
  So, the formula is simply: **nr  
  Permutations without Repetition:**(In this case, we have to reduce the number of available choices each time)  
  **Example**: **what order could 16 pool balls be in?**After choosing, say, number "14" we can't choose it again.  
  So, our first choice has 16 possibilities, and our next choice has 15 possibilities, then 14, 13, 12, 11, ... etc. And the total permutations are:  
   **16 × 15 × 14 × 13 × ... = 20,922,789,888,000**But maybe we don't want to choose them all, just 3 of them, and that is then: **16 × 15 × 14 = 3,360**



**How to do it mathematically ?**when we want to select all of the billiard balls the permutations are:  
 **16! = 20,922,789,888,000**But when we want to select just 3 we don't want to multiply after 14. How do we do that? There is a neat trick: we divide by 13!

****

That was neat. The 13 × 12 × ... etc gets "cancelled out", leaving only 16 × 15 × 14.  
**So we can calculate it using :**

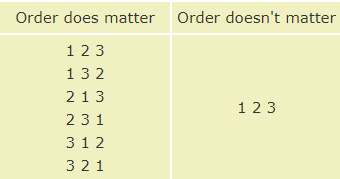


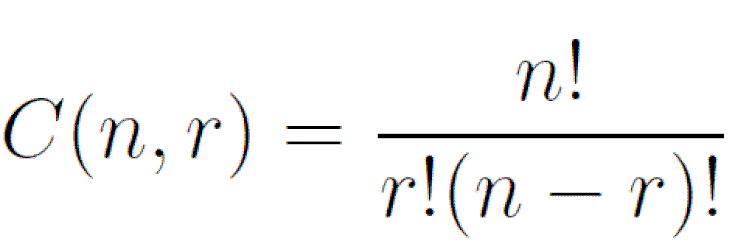
**Now how to do it in code ?!**

| int n,r,f1=1;  for(int i=2; i<=n; i++) *//calculating factorial of n*  {  f1 = f1\*i;  }  int f2 = 1;  for (int i=2; i<=(n-r); i++) *//calculating factorial of (n-r)*  {  f2 = f2\*i;   }  int ans = f1/f2; |
| --- |
|  |

* **Combination**  
  This is how lotteries work. The numbers are drawn one at a time, and if we have the lucky numbers (no matter what order) we win!  
  **The easiest way to explain it is to:**assume that the order does matter **( permutations**),  
  then alter it so the order does not matter.

Going back to our pool ball example, let's say we just want to know which 3 pool balls are chosen, not the order.  
We already know that 3 out of 16 gave us 3,360 permutations.  
1But many of those are the same to us now, because we don't care what order!

**For example, let us say balls 1, 2 and 3 are chosen. These are the possibilities:**

**So, the permutations have 6 times as many possibilities.  
You can notice that 6 is factorial of 3   
So to eliminate repeated from permutation we can divide by r!  
  
We can calculate it using following formula :**

**How to do it in code ?**

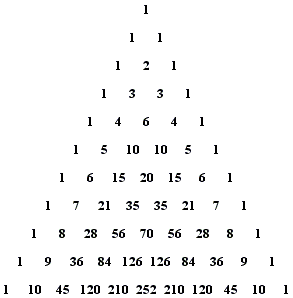
| **int n,r,f1=1;  for(int i=2; i<=n; i++) *//calculating factorial of n*  {  f1 = f1\*i;  }  int f2 = 1;  for (int i=2; i<=(n-r); i++) *//calculating factorial of (n-r)*  {  f2 = f2\*i;   }  int f3 = 1;  for(int i=2; i<=r; i++) *//calculating factorial of r*  {  f3 = f3\*i;  }  int ans = f1/(f3\*f2);** |
| --- |
|  |

But as you know, Factorial grows very fast which won’t help us calculate large numbers due to overflow   
As you can notice by dividing by r! cancels multiplies from 1 to r   
So we can multiply from n to r

**So here is an optimization:**

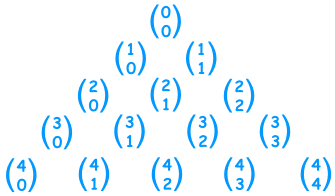
| long long n,r,res = 1, k = 1;  for (int i = n; i > r; i--) {  res \*= i;  res /= k;  k++;  } |
| --- |

**Pascal Triangle**

We can calculate combination using another interesting number Pattern called Pascal Triangle 

To build the triangle, start with "1" at the top, then continue placing numbers below it in a triangular pattern.

Each number is the numbers directly above it added together  
  
**How to get nCr ?**

You should look to row n ( where rows starts from 0 ) and cell r  


**How to do it in code ?**

| int arr[100][100];    for (int line = 0; line < 100; line++)  {   *// Every line has number of integers*   *// equal to line number*   for (int i = 0; i <= line; i++)   {   *// First and last values in every row are 1*   if (line == i || i == 0)   arr[line][i] = 1;   *// Other values are sum of values just*   *// above and left of above*   else   arr[line][i] = arr[line-1][i-1] + arr[line-1][i];   }  } |
| --- |

**Probability**

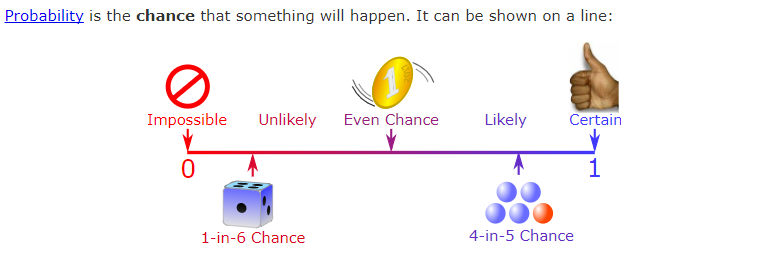
Probability is the mathematical term for the likelihood that something will occur, such as drawing an ace from a deck of cards or picking a green piece of candy from a bag of assorted colors. You use probability in daily life to make decisions when you don't know for sure what the outcome will be. Most of the time, you won't perform actual probability problems, but you'll use subjective probability to make judgment calls and determine the best course of action.



When a coin is tossed, there are two possible outcomes , We say that the chance of the coin landing **Head** is And the chance of the coin landing **Tail** is ½ .

So we Can define the chance of something happening as **event** and We can define an **event** as any collection of outcomes of an experiment. Thus, an event is a subset of the sample space S. If we denote an event by **E**, we could say that **E⊆S**. If an event consists of a single outcome in the sample space, it is called a simple event. Events which consist of more than one outcome are called compound events.

What we are actually interested in is the probability of a certain event to occur, or **P(E)**. By definition, **P(E)** is a real number between 0 and 1, where 0 denotes the impossible event and 1 denotes the certain event (or the whole sample space).



In general probability of event happening = Number of ways it can happen

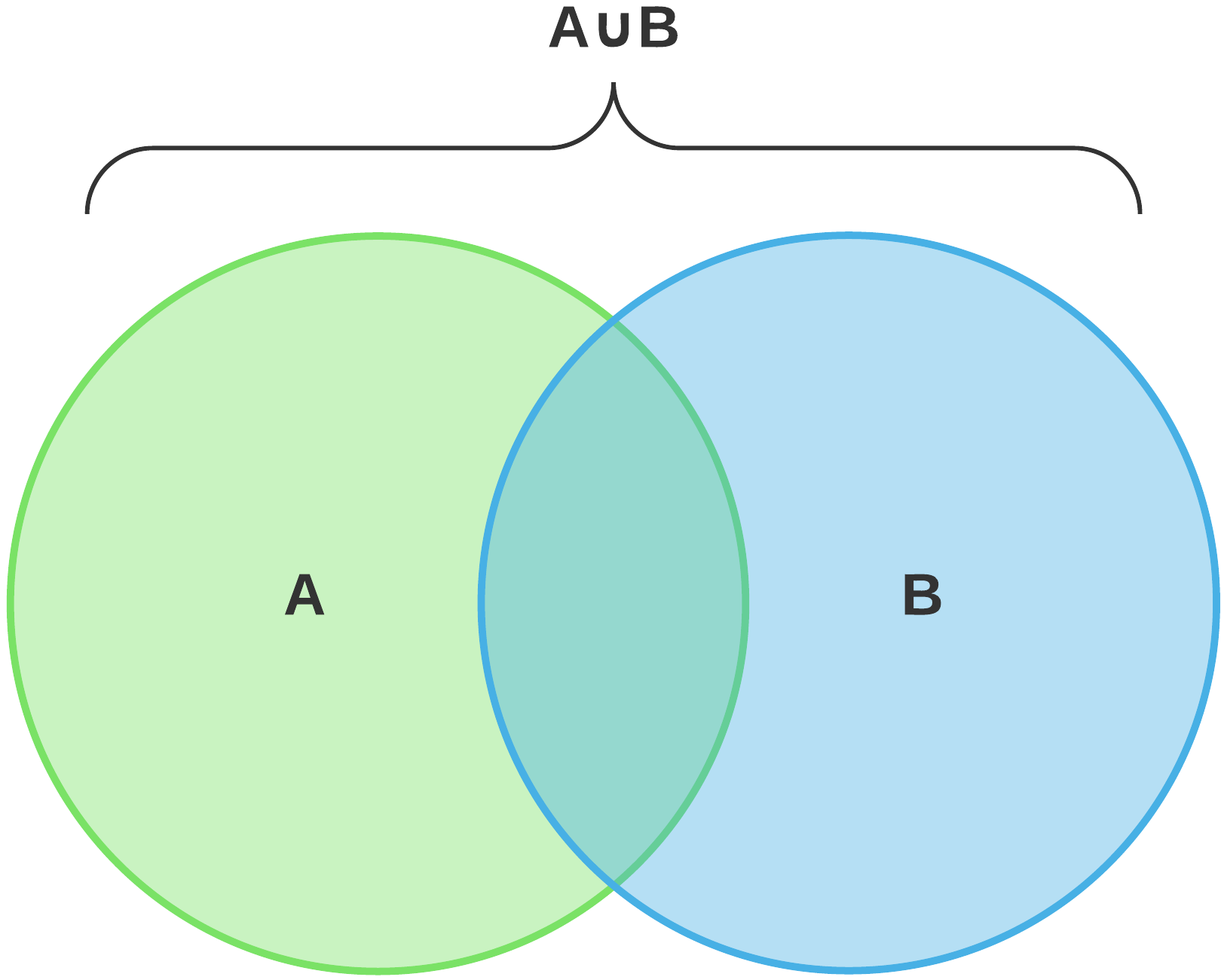


Total number of outcomes

**Union**

The union of two sets is a new set that contains all of the elements that are in at least one of the two sets. The union is written as A∪B or “A or B”.

A ∪ B means ”A or B happen”. For example, the union of A = {2,5} and B = {4,5,6} is A ∪B = {2,4,5,6}.

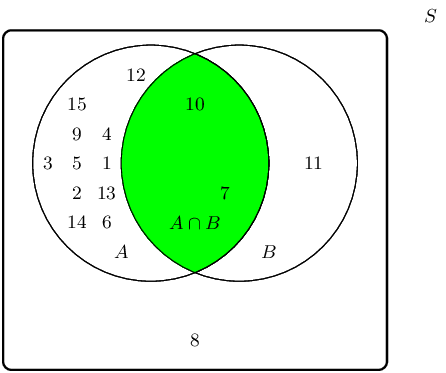
****

**Intersection**

The intersection of two sets is a new set that contains all of the elements that are in both sets. The intersection is written as A∩B or “A and B”.

Intersection is the set of all distinct elements that are in both and . We define the intersection of a collection of sets, as the set of all distinct elements that are in all of these sets.

The intersection between event set A and event set B, A∩B, can be shaded as follows:

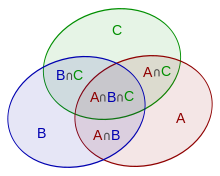


Therefore the event set {7;10} best describes the event set of A∩B.

**Inclusion–exclusion:** principle is a counting technique which generalizes the familiar method of obtaining the number of elements in the union of two finite sets; symbolically expressed as |A∪B|=|A|+|B|-|A∩B|.

where *A* and *B* are two finite sets and |*S*| indicates the cardinality of a set *S* (which may be considered as the number of elements of the set, if the set is finite). The formula expresses the fact that the sum of the sizes of the two sets may be too large since some elements may be counted twice. The double-counted elements are those in the intersection of the two sets and the count is corrected by subtracting the size of the intersection.

The principle is more clearly seen in the case of three sets, which for the sets *A*, *B* and *C* is given by |A|+|B|+|C|-(|A∩B|+|A∩C|+|B∩C|)+(|A∩B∩C|)



This formula can be verified by counting how many times each region in the Venn diagram above is included in the right-hand side of the formula. In this case, when removing the contributions of over-counted elements, the number of elements in the mutual intersection of the three sets has been subtracted too often, so must be added back in to get the correct total.