Bell Numbers (Number of ways to Partition a Set)

Count number of ways to partition a set into k subsets (sub-question)

Let S(n, k) be total number of partitions of n elements into k sets.

$$S(n,k) = k * S(n-1,k) + S(n-1,k-1)$$

When we add a nth element to k partitions, there are two possibilities

1. It is added as a single element set to existing partitions, i.e, S(n, k-1)
2. It is added to all sets of every partition, i.e., k*S(n, k)

S(n, k) is called `Stirling numbers of the second kind`

```
#include<iostream>
using namespace std;

// Returns count of different partitions of n
// elements in k subsets
int countP(int n, int k)
{
  // Base cases
  if (n == 0 || k == 0 || k > n)
     return 0;
  if (k == 1 || k == n)
      return 1;

  // S(n+1, k) = k*S(n, k) + S(n, k-1)
  return  k*countP(n-1, k) + countP(n-1, k-1);
}

// Driver program
int main()
{
   cout <<  countP(3, 2);
   return 0;
```

What is a Bell Number?

The value of nth Bell Number is sum of S(n, k) for k = 1 to n.

$$Bell(n) = \sum_{k=0}^{n} S(n, k)$$

Compute Bell Number

A Simple Method to compute nth Bell Number is to one by one compute S(n, k) for k = 1 to n and return sum of all computed values.

A Better Method is to use `Bell Triangle`

Bell Triangle

Below is a sample Bell Triangle for first few Bell Numbers

```
1
1 2
2 3 5
5 7 10 15
15 20 27 37 52
```

The triangle is constructed using below formula

```
// If this is first column of current row 'i'
If j == 0
   // Then copy last entry of previous row
   // Note that i'th row has i entries
   Bell(i, j) = Bell(i-1, i-1)

// If this is not first column of current row
Else
   // Then this element is sum of previous element
   // in current row and the element just above the
   // previous element
   Bell(i, j) = Bell(i-1, j-1) + Bell(i, j-1)
```

```
// A C++ program to find n'th Bell number
#include<iostream>
using namespace std;

int bellNumber(int n)
{
   int bell[n+1][n+1];
   bell[0][0] = 1;
   for (int i=1; i<=n; i++)
   {
      // Explicitly fill for j = 0
      bell[i][0] = bell[i-1][i-1];
```

```
      // Fill for remaining values of j
      for (int j=1; j<=i; j++)
         bell[i][j] = bell[i-1][j-1] + bell[i][j-1];
   }
   return bell[n][0];
}

// Driver program
int main()
{
   for (int n=0; n<=5; n++)
      cout << "Bell Number " << n << " is "
           << bellNumber(n) << endl;
   return 0;
}
```