Permutation Coefficient

排列数的计算

$$P(n, k) = n \cdot (n-1) \cdot (n-2) \ldots (n-k+1)$$

O(n *k)

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

```
int permutationCoeff(int n, int k)
{
    int P[n + 1][k + 1];

    // Caculate value of Permutation
    // Coefficient in bottom up manner
    for (int i = 0; i <= n; i++)
    {
        for (int j = 0; j <= std::min(i, k); j++)
        {
            // Base Cases
            if (j == 0)
                P[i][j] = 1;

            // Calculate value using
            // previosly stored values
            else
                P[i][j] = P[i - 1][j] +
                          (j * P[i - 1][j - 1]);

            // This step is important
            // as P(i,j)=0 for j>i
            P[i][j + 1] = 0;
        }
    }
    return P[n][k];
}
```

O(n)

采用1维数组来进行计算 n!，利用如下公式得出结果

$$P(n, k) = n! / (n-k)!$$

```
int permutationCoeff(int n, int k)
{
    int fact[n + 1];

    // base case
    fact[0] = 1;

    // Caculate value
    // factorials up to n
    for (int i = 1; i <= n; i++)
        fact[i] = i * fact[i - 1];

    // P(n,k) = n! / (n - k)!
    return fact[n] / fact[n - k];
}
```

A O(n) time and O(1) Extra Space Solution

```
int PermutationCoeff(int n, int k)
{
    int Fn = 1, Fk;

    // Compute n! and (n-k)!
    for (int i = 1; i <= n; i++)
    {
        Fn *= i;
        if (i == n - k)
        Fk = Fn;
    }
    int coeff = Fn / Fk;
    return coeff;
}
```