数字串相乘

问题描述：设有一个长度N的数字串，要求选手使用K个乘号将它分成K+1个部分，找出一种分法，使得这K+1个部分的乘积能够为最大。

最优解结构特征

假设求前n个数最大乘积，分割次数为k，则最大乘积一定为，前n-j个数分割k-1次取得的最大值*第j+1到n的值的最大值，其中j的取值范围为[k, n-1]。

定义最优解

$$dp[i,k] = \begin{cases} a[j] & i = 0 \\ \max_{k \le j < i} \{dp[j, k-1] * num[j+1, j]\} & i < j \end{cases}$$

其中a为输入的数字串，k代表分割的次数，j代表在第j个点进行分割。

计算最优解

```
int array_to_num(vector<int> &num){
    int data = 0;
    for (int i = 0; i < num.size(); i++) {
        data = data * 10 + num[i];
    }
    return data;
}

vector<vector<int>> maximun_product(vector<vector<int>> &dp, vector<int> &num, int k){
    int max = 0;
    for (int k_count = 1; k_count <= k; k++) {
        for (int i = k_count + 1; i < num.size(); i++) {
            for (int j = k_count; j < i; j++) {
                vector<int> data(num.begin()+ j + 1, num.begin() + i);
                int result = dp[j][k_count - 1] * array_to_num(data);
                if (max < result) {
                    max = result;
                }
            }
            dp[i][k_count] = max;
        }
    }
    return dp;
}
```

构造最优解

若要确定分割点，则需要将每次最大值的位置记录下来，再进行寻找。