算法分析题

2-2 7 个二分搜索算法

下面的 7 个算法与本章中的二分搜索算法 BinarySearch 略有不同。请判断这 7 个算法的正确性。如果算法不正确，请说明产生错误的原因。如果算法正确，请给出算法的正确性证明。

-
```
template<class Type>
int BinarySearch1(Type a[ ], const Type& x, int n)
    {
    int left=0; int right=n-1;
    while (left <= right){
      int middle = (left+right)/2;
      if (x == a[middle]) return middle;
       if (x > a[middle]) left = middle;
      else right = middle;
      }
    return -1;
    }
```
-
```
template<class Type>
int BinarySearch2(Type a[ ], const Type& x, int n)
{
 int left=0; int right=n-1;
while (left < right-1){
      int middle = (left+right)/2;
      if (x < a[middle]) right = middle;
      else left = middle;
      }
if (x == a[left])    return left;
 return -1;
}
```
-
```
template<class Type>
int BinarySearch3(Type a[ ], const Type& x, int n)
{
int left=0; int right=n-1;
```

```
while (left +1!= right){
  int middle = (left+right)/2;
  if (x >= a[middle]) left = middle;
  else right = middle;
  }
if (x == a[left])    return left;
return -1;
}
```

- ```
  template<class Type>
  int BinarySearch4(Type a[ ], const Type& x, int n)
  {
  if(n>0 && x>=a[0]){
  int left=0; int right=n-1;
  while (left < right){
    int middle = (left+right)/2;
    if (x < a[middle]) right = middle-1;
    else left = middle;
    }
  if (x == a[left])    return left;
  }
  return -1;
  }
  ```

- ```
  template<class Type>
  int BinarySearch5(Type a[ ], const Type& x, int n)
  {
  if(n>0 && x>=a[0]){
  int left=0; int right=n-1;
  while (left < right){
    int middle = (left+right+1)/2;
    if (x < a[middle]) right = middle-1;
    else left = middle;
    }
  if (x == a[left])    return left;
  }
  ```

```
return -1;
}
```

- ```
  template<class Type>
  ```

```
int BinarySearch6(Type a[ ], const Type& x, int n)
{
if(n>0 && x>=a[0]){
int left=0; int right=n-1;
while (left < right){
  int middle = (left+right+1)/2;
  if (x < a[middle]) right = middle-1;
  else left = middle+1;
  }
if (x == a[left])    return left;
}
return -1;
}
```

- ```
  template<class Type>
  ```

```
int BinarySearch7(Type a[ ], const Type& x, int n)
{
if(n>0 && x>=a[0]){
int left=0; int right=n-1;
while (left < right){
  int middle = (left+right+1)/2;
  if (x < a[middle]) right = middle;
  else left = middle;
  }
if (x == a[left])    return left;
}
return -1;
}
```

2-7 多项式乘积

设 $P(x)=a_0+a_1x+\cdots+a_dx^d$ 是一个 d 次多项式。假设已有一个算法能在 $O(i)$ 时间内计算一个 i 次多项式与一个一次多项式的乘积，以及一个算法能在 $O(i\log i)$ 时间内计算两个 i 次多项式的乘积。对于任意给定的 d 个整数 $n_1,n_2,\cdots n_d$ ，用分治法设计一个有效算法，计算出满足 $P(n_1)=P(n_2)=\cdots=P(n_d)=0$ 且最高次项系数为 1 的 d 次多项式 $P(x)$ ，并分析算法的效率。

2-34 构造 Gray 码的分治算法

Gray码是一个长度为 2^n 的序列。序列中无相同元素，每个元素都是长度为n位的（0,1）串，相邻元素恰好只有一位不同。用分治策略设计一个算法对任意的 n 构造相应的 Gray 码。

算法实现题

2-14 整数因子分解问题

- **问题描述：**

  大于 1 的正整数 n 可以分解为：$n=x_1*x_2*\cdots*x_m$。

  例如，当 n=12 时，共有 8 种不同的分解式：

  12=12；

  12=6*2；

  12=4*3；

  12=3*4；

  12=3*2*2；

  12=2*6；

  12=2*3*2；

  12=2*2*3。

- **编程任务：**

  对于给定的正整数 n，计算 n 共有多少种不同的分解式。

- **数据输入：**

  由文件 input.txt 给出输入数据。第一行有 1 个正整数 n（$1\le n\le 2000000000$）。

- **结果输出：**

  将计算出的不同分解式数输出到文件 output.txt。

输入文件示例	输出文件示例
input.txt	output.txt
12	8