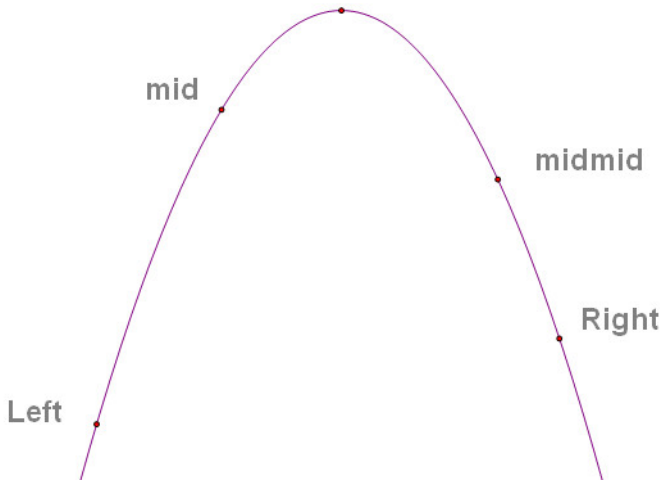
一. 概念

在二分查找的基础上，在右区间（或左区间）再进行一次二分，这样的查找算法称为三分查找，也就是三分法。
三分查找通常用来迅速确定最值。

二分查找所面向的搜索序列的要求是：具有单调性（不一定严格单调）；没有单调性的序列不是使用二分查找。
与二分查找不同的是，三分法所面向的搜索序列的要求是：序列为一个凸性函数。通俗来讲，就是该序列必须有一个最大值（或最小值），在最大值（最小值）的左侧序列，必须满足不严格单调递增（递减），右侧序列必须满足不严格单调递减（递增）。如下图，表示一个有最大值的凸性函数：



二、算法过程

（1）、与二分法类似，先取整个区间的中间值mid。

[cpp]view plaincopy

```
1. mid = (left + right) / 2;
```

（2）、再取右侧区间的中间值midmid，从而把区间分为三个小区间。

[cpp]view plaincopy

```
1. midmid = (mid + right) / 2;
```

（3）、我们mid比midmid更靠近最值，我们就舍弃右区间，否则我们舍弃左区间？。
比较mid与midmid谁最靠近最值，只需要确定mid所在的函数值与midmid所在的函数值的大小。当最值为最大值时，mid与midmid中较大的那个自然更为靠近最值。最值为最小值时同理。

[cpp]view plaincopy

```
1. if (cal(mid) > cal(midmid))
2.     right = midmid;
3. else
4.     left = mid;
```

（4）、重复（1）（2）（3）直至找到最值。

（5）、另一种三分写法

[cpp]view plaincopy

```
1. double three_devide(double low,double up)
2. {
3. double m1,m2;
4. while(up-low>=eps)
```

```
    {
        m1=low+(up-low)/3;
        m2=up-(up-low)/3;
if(f(m1)<=f(m2))
            low=m1;
else
             up=m2;
    }
return (m1+m2)/2;
}
```

算法的正确性：

1、mid与midmid在最值的同一侧。由于凸性函数在最大值（最小值）任意一侧都具有单调性，因此，mid与midmid中，更大（小）的那个 数自然更为靠近最值。此时，我们远离最值的那个区间不可能包含最值，因此可以舍弃。

2、mid与midmid在最值的两侧。由于最值在中间的一个区间，因此我们舍弃一个区间后，并不会影响到最值

[cpp]view plaincopy

```
const double EPS = 1e-10;
double calc(double x)
{
// f(x) = -(x-3)^2 + 2;
return -(x-3.0)*(x-3.0) + 2;
}
double ternarySearch(double low, double high)
{
double mid, midmid;
while (low + EPS < high)
    {
        mid = (low + high) / 2;
        midmid = (mid + high) / 2;
double mid_value = calc(mid);
double midmid_value = calc(midmid);
if (mid_value > midmid_value)
            high = midmid;
else
            low = mid;
    }
return low;
}
```

调用ternarySearch(0, 6)，返回的结果为3.0000