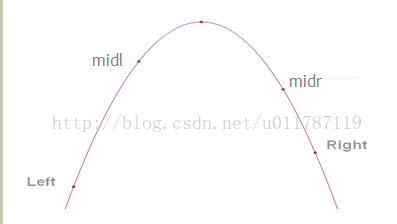
# 三分法

类似于二分查找，三分搜索法也是比较常用的基于分治思想的高效查找方法。但是和二分不同，二分只适用于单调函数，比如常用的对单调递增或单调递减的一个序列中的某一个元素进行查找，三分却突破了这种限制，可以用于左边递增右边递减或者相反的，这么一类函数，也就是常说的凸函数和凹函数。但是为什么三分法可以用于凸函数或者凹函数呐，这其实是因为这种函数总是有一个最大值或者最小值，这样就可以借此判断出三分法中两个中点相对相对于极值的位置，例如下图（凹函数类似）：

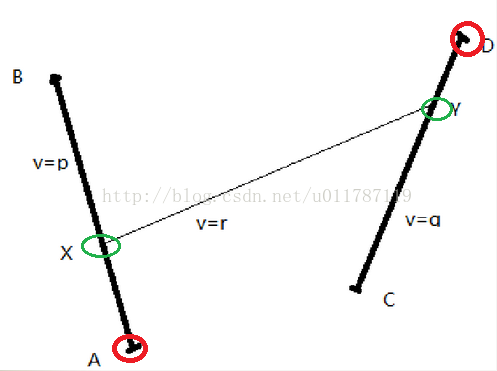


三分搜索的实现主要是判断midl和midr所在值的大小。以凸函数为例（凹函数类似，只是判mid大小的时候保留小的即可（其实也是保留离极值最近的mid）），先以left和right为端点计算出它们的中点midl，然后再以midl和right为端点计算出它们的中点midr，接下来就需要判断f(midl)和f(midr)值的大小了，如果f(midl)大于f(midr)，那么说明midl靠近极值，此时令right=midr，否则说明midr靠近极值，此时则令left=midl，总之就是要保留离极值最近的那一个mid，然后重复前面的过程，直到left和right十分接近，最终f(left)就等于了极值，下面给出程序实现：

1. **double** solve(**double** parameter)
2. {
3. // 计算函数值，即f(x)
4. }
6. **double** trisection\_search(**double** left, **double** right)
7. {
8. // 三分搜索，找到最优解（求函数最大值下的自变量值）
9. **double** midl, midr;
10. **while** (right-left > 1e-7)
11. {
12. midl = (left + right) / 2;
13. midr = (midl + right) / 2;
14. // 如果是求最小值的话这里判<=即可
15. **if**(solve(midl) >= solve(midr)) right = midr;
16. **else** left = midl;
17. }
18. **return** left;
19. }

Hdu 3400

题意：



**题意**：   
就是给你两条线段AB , CD ，一个人在AB以速度p跑,在CD上以q跑,在其他地方跑速度是r。问你从A到D最少的时间。

**分析**：   
先三分AB上的点，再三分CD上的点即可。   
证明：   
设E在AB上，F在CD上。   
令人在线段AB上花的时间为：f = AE / p，人走完Z和Y所花的时间为：g = EF / r + FD / q。   
f函数是一个单调递增的函数，而g很明显是一个先递减后递增的函数。两个函数叠加，所得的函数应该也是一个先递减后递增的函数。故可用三分法解之。