


二分和分治思想

By Fancy


二分查找的应用条件

- 单调性!
  - 二分查找通常应用于具有单调性的数列（离散值）或函数（连续值）
  - 数列的单调性通常指的是随着数列下标的增长，数列元素的值不增或不减，例如0, 1, 1, 2, 3, 5, 8
  - 函数f(x)的单调性通常是这样定义的：对于任意的x0<x1，f(x0)>=f(x1)

这里数列的例子是不减的，函数定义的例子是不增的

离散值的二分查找

- 通常我们采用结构如下的算法实现离散值的二分查找：

```
l = 0; r = range;
while (l < r) {
    mid = (l + r) / 2;
    if (val[mid] < request)
        l = mid + 1;
    else
        r = mid;
}
```

这里的range是离散值的下标范围，val是通过下标取得离散值的数组，而request是我们需要查找的值
最终l储存的是第一个大于等于request的值的下标


离散值的二分查找

- 例：在0, 1, 1, 2, 3, 5, 8, 13, 21中查找4
- 例：在100, 90, 80, 70, 60, 50, 10中查找90


STL中的二分查找

```
// Binary search (lower_bound, upper_bound, equal_range, binary_search).

template <class _ForwardIter, class _Tp, class _Distance>
_ForwardIter __lower_bound(_ForwardIter __first, _ForwardIter __last,
                           const _Tp& __val, _Distance*)
{
  _Distance __len = 0;
  distance(__first, __last, __len);
  _Distance __half;
  _ForwardIter __middle;

  while (__len > 0) {
    __half = __len >> 1;
    __middle = __first;
    advance(__middle, __half);
    if (*__middle < __val) {
      __first = __middle;
      ++__first;
      __len = __len - __half - 1;
    }
    else
      __len = __half;
  }
  return __first;
}
```


连续值的二分查找

- 常用的连续值二分查找过程：

```
l = range_low; r = range_high;
calculate_time = 0;
while (calculate_time < max_time) {
    mid = (l + r) / 2;
    if (f(mid) < request)
        l = mid;
    else
        r = mid;
    calculate_time++;
}
```

这里的range_low和range_high是函数的定义域或可能对应f(x)=request的范围，calculate_time统计二分的次数，max_time是二分次数的上限，f是表示连续值的函数，而request是我们需要查找的值最终f(l) = request


连续值的二分查找

- 例：令f(x)=3^x，求f(x)值为100时相应的x


分治

- 二分实际上是将问题的规模不断缩小到原来的一半并最终求得结果的方法；
- 而分治并不一定是将规模缩小至一半的方法，它是将问题不断分为两份或多份，通过分别解决每一部分并最终合并求得结果


分治

- kth element：在一个无序表中查找第k大的元素
- 快速排序
- 线段树
- 动态规划的优化


扩展阅读

- 顾昱洲 浅谈一类分治算法
- 陈丹琦 从《Cash》谈一类分治算法的应用
- http://zh.wikipedia.org/wiki/偏序关系


作业

- HW3-1
- HW3-2