41 专题3：二分算法

更新时间：2019-10-14 09:25:20

> 我要扼住命运的咽喉，它妄想使我屈服，这绝对办不到。生活是这样美好，活他一千辈子吧！
>
> ——贝多芬

最近在挺多公司的笔试题中，看到了关于二分算法使用的面试题，正好在专栏的最后一部分总结一下这个算法。二分算法属于是比较简单的算法思想，代码实现简洁。但是比较考验人的思维能力，也是笔试题中比较常见的题型。

什么是二分算法？

在计算机科学中，二分搜索（binary search），也称折半搜索（half-interval search）、对数搜索（logarithmic search），是一种在**有序数组**中查找某一特定元素的搜索算法。搜索过程从数组的中间元素开始，如果中间元素正好是要查找的元素，则搜索过程结束；如果某一特定元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半中查找，而且跟开始一样从中间元素开始比较。如果在某一步骤数组为空，则代表找不到。这种搜索算法每一次比较都使搜索范围缩小一半。

二分的模型

- 给定一个递增（递减）数列，检索是否存在等于m的元素
- 给定一个单调函数f(x)，检索 f(x) <= m 的最大x值
- 给定一个单调函数f(x)，检索 f(x) >= m 的最小x值

二分的关键词是，***单调（有序）***

二分的例子

下面我们来看几道具体的面试题，来看下二分算法具体是要怎么运用的。

例子1：（2019.09.04 携程笔试题）

> 在m个节点的分布式计算系统中，有一批任务需要执行，每个任务需要的时间是array[i]，每个节点同一时间只能执行一个任务，每个节点只能执行连续的任务，例如 i，i+1，i+2，但是不能执行 i，i+2， 请问任务完成的最短时间。

解析：

这题可以用动态规划做，令 dp[i][j] 表示前i个任务，使用 j 个节点，完成的最短时间，于是我们有状态转移方程：

dp[i][j] = min(dp[k][j-1] + sum(array[k + 1], ... , array[i]))，求dp[n][m]，复杂度是o(n^2*m)。

如果按照 leetcode-410 的数据范围，n<=1000，m<=min(50，n)，那么动态规划的解法可以过，请大家有兴趣可以使用动态规划解 leetcode-410。但如果 n<=100000 呢？

m个节点跟任务完成最短的时间t的关系是m越小，t越长，m越大，t越小。令f(t)为任务完成时间为t，需要的节点数。f(t)是个单调递减函数。按照上面说的，符合二分的模型，检索f(t)<=m的最小t值（因为函数是单调递减函数，所以跟上面的二分模型是反过来的）

还剩一个问题，f(t)怎么求？贪心！第一个节点从第1个任务开始执行起，由于任务只能连续的执行，第一个节点执行完第1个任务之后，尽可能往后执行更多的任务，直到任务时间和***正好***不超过t。第二个节点也是如此。循环到最后一个任务，可以得到f(t)的值，就是任务完成时间为t的情况下，需要多少个节点。

二分的上下界

下界：假设节点数无限大，那么每个节点只完成一个任务，任务完成的时间就是所有任务需要时间的最大值；

上界：假设只有一个节点，那么任务完成时间就是所有任务需要时间之和。

二分的复杂度？

o(log(上界-下界)*n)

例子2（2019.09.06 云从科技笔试题）

> 农场主鲍勃建造了一座有n间羊圈的房子，羊圈排在一条直线上，第i间羊圈在x[i]的位置，但是鲍勃的m只羊对小屋很不满意，因此经常打架，鲍勃为了防止羊之间互相伤害，因此决定把每只羊都放在离其他羊尽可能远的羊圈，也就是要最大化最近的两只羊之间的距离。
> 羊羊并不喜欢这种布局，而且几只羊放在一个隔间里，它们就要发生争斗，为了不让羊互相伤害，鲍勃决定自己给羊分配隔间，使任意两只羊的最小距离尽可能的大，那么，这个最大的最小距离是多少呢？

解析：

看过上一题，再看这一题就没什么难度了。模型基本一样，都是求 f(t)>=n 的最小值。

我们假设羊之间的最小距离是 t，在所有羊圈中安排羊的住处。从最左边的第一个羊圈开始，选x[0]，再往后寻找比 x[0] 至少大t的羊圈，作为第二只羊的住处。这样循环寻找下去，我们就可以知道整个羊圈能放多少只羊。

显然，距离 t 越大，整个羊圈能放的羊就越少，所以f(t)是单调递减的。

由于题目要求，羊圈要放m只羊，所以我们需要求f(t)>=m的t的最大值。这也是一个经典的二分使用场景。

例子3

（leetcode-1011 https://leetcode-cn.com/problems/capacity-to-ship-packages-within-d-days）

> 传送带上的包裹必须在 D 天内从一个港口运送到另一个港口。
>
> 传送带上的第 i 个包裹的重量为 weights[i]。每一天，我们都会按给出重量的顺序往传送带上装载包裹。我们装载的重量不会超过船的最大运载重量。
>
> 返回能在 D 天内将传送带上的所有包裹送达的船的最低运载能力。

解析：

假设最大运载重量是t，第一天，从第一个包裹开始，尽可能装载更多的包裹，直到重量和到达t，第二天也是如此。这样循环下去，我们可以知道运送完所有的包裹需要多少天。

显然，t越大，时间越短，所以f(t)是单调递减的。

题目要求在D天内运送完，所以还是求f(t)<=D的t的最小值。

二分的模板

我用 Python 给大家总结了一个二分算法的模板，可以直接拿来用。

- 在一个有序的数组中寻找某个值：

```
int bsearch(int* A, int n, int k) {
    int left = -1, right = n;
    while (right - left > 1) {
        int mid = (left + right) / 2;
        if (A[mid] == k) {
            return mid;
        } else if (A[mid] < k) {
            left = mid;
        } else {
            right = mid;
        }
    }
    return -1;
}
```

- 满足judge(x)=true的最大值：

```
bool judge(int x) {}
int bsearch(int min, int max) {
    int left = min, right = max;
    //这里为什么要right++呢，因为我们求的是上界，二分区间中，left要满足f(x)，并且尽可能大
    //需要要保证left在[min, max]遍历到
    //终止条件是right - left > 1，所以right需要+1确保left能取到min
    right++;
    while (right - left > 1) {
        int mid = (left + right) / 2;
        if (judge(mid)) {
            left = mid;
        } else {
            right = mid;
        }
    }
    return left;
}
```

- 满足judge(x)=true的最小值

```
bool judge(int x) {}
int bsearch(int min, int max) {
    int left = min, right = max;
    //这里为什么要left--呢，因为我们求的是下界，二分区间中，right要满足f(x)，并且尽可能小
    //需要要保证right在[min, max]遍历到
    //终止条件是right - left > 1，所以left需要-1确保right能取到max
    left--;
    while (right - left > 1) {
        int mid = (left + right) / 2;
        if (judge(mid)) {
            right = mid;
        } else {
            left = mid;
        }
    }
    return right;
}
```

- 浮点数，求f(x)在[min,max]的零点：

```
const double EPS = 1e-8;
double cal(double x) {} //假定是单调增
//这里需要保证区间有零点
double bsearch(double min, double max) {
    double left = min, right = max;
    while (right - left > EPS) {
        double mid = (left + right) / 2;
        if (A[mid] == 0) {
            return mid;
        } else if (cal(mid) < 0) {
            left = mid;
        } else {
            right = mid;
        }
    }
    return (left + right) / 2;
}
```

总结

这一节主要是讲了二分算法的原理和一些使用场景，也带着大家看了两道公司的面试题目，最后给大家总结了一个二分算法的使用模板。但是只有 Python 的，如果你的常用语言不是 Python 的话，也可以参考 Python 的模板来用你自己的语言实现一下。

}

40 专题2：二叉树遍历

42 专题4：斐波那契数列