▶

note 49 views

July 16 BinarySearchModels

```
//  非严格递增
//  模板1     f(a[pos]) <= target < f(a[pos + 1])
public int func(input...) {
    //  经过分析，将原问题转为
    //  白板上的东西
    T[] arr = new T[n];
    G target = xxx;
    int start = 0;
    int end = n - 1;
    int pos = -1;
    while (start <= end) {
        int mid = start + (end - start) / 2;
        G midVal = g(mid);
        if (midVal <= target) {
            pos = mid;
            start = mid + 1;
        } else {
            end = mid - 1;
        }
    }
    return pos;
}
```

```
//  非严格递增
//  模板2     f(a[pos]) < target <= f(a[pos + 1])
public int func(input...) {
    //  经过分析，将原问题转为
    //  白板上的东西
    T[] arr = new T[n];
    G target = xxx;
    int start = 0;
    int end = n - 1;
    int pos = -1;
    while (start <= end) {
        int mid = start + (end - start) / 2;
        G midVal = g(mid);
        if (midVal < target) {
            pos = mid;
            start = mid + 1;
        } else {
            end = mid - 1;
        }
    }
    return pos;
}
```

```
//  非严格递增
//  模板3     f(a[pos - 1]) <= target < f(a[pos])
public int func(input...) {
    //  经过分析，将原问题转为
    //  白板上的东西
    T[] arr = new T[n];
    G target = xxx;
    int start = 0;
    int end = n - 1;
    int pos = n;
    while (start <= end) {
        int mid = start + (end - start) / 2;
        G midVal = g(mid);
        if (midVal > target) {
            pos = mid;
            end = mid - 1;
        } else {
            start = mid + 1;
        }
    }
    return pos;
}
```

```
//  非严格递增
//  模板4     f(a[pos - 1]) < target <= f(a[pos])
public int func(input...) {
    //  经过分析，将原问题转为
    //  白板上的东西
    T[] arr = new T[n];
    G target = xxx;
    int start = 0;
    int end = n - 1;
    int pos = n;
    while (start <= end) {
        int mid = start + (end - start) / 2;
        G midVal = g(mid);
        if (midVal >= target) {
            pos = mid;
            end = mid - 1;
        } else {
            start = mid + 1;
        }
    }
    return pos;
}
```

```
//  非严格递减
//  模板5     f(a[pos]) >= target > f(a[pos + 1])
public int func(input...) {
    //  经过分析，将原问题转为
    //  白板上的东西
    T[] arr = new T[n];
    G target = xxx;
    int start = 0;
    int end = n - 1;
    int pos = -1;
    while (start <= end) {
        int mid = start + (end - start) / 2;
        G midVal = g(mid);
        if (midVal >= target) {
            pos = mid;
            start = mid + 1;
        } else {
            end = mid - 1;
        }
    }
    return pos;
}
```

```
//  非严格递减
//  模板6     f(a[pos]) > target >= f(a[pos + 1])
public int func(input...) {
    //  经过分析，将原问题转为
    //  白板上的东西
    T[] arr = new T[n];
    G target = xxx;
    int start = 0;
    int end = n - 1;
    int pos = -1;
    while (start <= end) {
        int mid = start + (end - start) / 2;
        G midVal = g(mid);
        if (midVal > target) {
            pos = mid;
            start = mid + 1;
        } else {
            end = mid - 1;
        }
    }
    return pos;
}
```

```java
//  非严格递减
//  模板7     f(a[pos - 1]) >= target > f(a[pos])
public int func(input...) {
    //  经过分析，将原问题转为
    //  白板上的东西
    T[] arr = new T[n];
    G target = xxx;
    int start = 0;
    int end = n - 1;
    int pos = n;
    while (start <= end) {
        int mid = start + (end - start) / 2;
        G midVal = g(mid);
        if (midVal < target) {
            pos = mid;
            end = mid - 1;
        } else {
            start = mid + 1;
        }
    }
    return pos;
}
```

```java
//  非严格递减
//  模板8     f(a[pos - 1]) > target >= f(a[pos])
public int func(input...) {
    //  经过分析，将原问题转为
    //  白板上的东西
    T[] arr = new T[n];
    G target = xxx;
    int start = 0;
    int end = n - 1;
    int pos = n;
    while (start <= end) {
        int mid = start + (end - start) / 2;
        G midVal = g(mid);
        if (midVal <= target) {
            pos = mid;
            end = mid - 1;
        } else {
            start = mid + 1;
        }
    }
    return pos;
}
```

八个模板如上，递增情况下，当midVal > target时，end = mid - 1;，砍掉后头到前面值更小的区间内找target，当midVal < target时，砍掉前头到后面值更大的区间内找target。递减情况下，当midVal > target时，start = mid + 1，砍掉前头到后面值更小的区间找target，当midVal < target时，砍掉后头到前面值更大的区间找target。pos的更新永远跟着if（condition）后面走。
如有错误，欢迎指正。谢谢大家。

binary_search

Updated 8 days ago by Chenhui Li and New Soft Valley

followup discussions *for lingering questions and comments*

[✓ Resolved] Unresolved

New Soft Valley 9 days ago

2，3，6，7不对

if (condition) {
  pos = mid;
  调整start或者end
} else {
  调整start或者end
}

pos = mid永远发生在if里面。不会在else里面。

Chenhui Li 9 days ago 终于理解了，还是昨晚说的那个问题，把要求什么搞混了，要求的是什么，if（condition）就写什么，pos = mid不变，然后start end根据砍前面还是后面来调整。

New Soft Valley 9 days ago
那快更新啊！这么多双眼睛盯着你们的代码！

New Soft Valley 9 days ago
第5，6的pos，应该是-1

另外每个模板里，
把
T[] arr = new T[size];
改写成
T[] arr = new T[n];

把
int end = arr.length - 1;
改成
int end = n - 1;

会方便以后查看一些

Bre 8 days ago 所有模版都调对吗？好像chenhui 还改了。

Chenhui Li 8 days ago 不好意思，刚刚改完了。

New Soft Valley 8 days ago

```
Ship-it!
```