Advanced Binary Search

- Srivaths P

Profiles: https://sriv.bio.link/

Implementation

Finds the last index of target

```
int search(vector<int> a, int target) {
    int left = 0, right = a.size() - 1;

    while (left < right) {
        int mid = (left + right + 1) / 2;

        if (a[mid] <= target) left = mid;
        if (a[mid] > target) right = mid - 1;
    }

    return (a[left] == target) ? left : -1;
}
```

Implementation

Finds the first index of target

```
int search(vector<int> a, int target) {
    int left = 0, right = a.size() - 1;

    while (left < right) {
        int mid = (left + right) / 2;

        if (a[mid] < target) left = mid + 1;
        if (a[mid] >= target) right = mid;
    }

    return (a[left] == target) ? left : -1;
}
```

Binary Search points

- = should be on the side which we should get rid of if a[mid] is = to target

- = side will be unchanging, and other will be changing

- Floor (left) or Ceil (right): Should lean towards the "changing" (var != mid) side

Binary Search Conditions

Binary search works on a set of elements where the “predicate” function applied on it is as follows:

$$T\ T\ T\ \ldots T\ T\ F\ F\ \ldots F\ F\ F$$

Binary search will move:

- L to mid when predicate is true (gets rid of left side of space).
- R to mid when predicate is false (gets rid of right side of space).

Alternative Binary Search

```
int l = min-1, r = max+1;
while (r-l > 1) {
    int m = (l + r) / 2;
    if (predicate(m))
        l = m;
    else
        r = m;
}

// l is the last true
// r is the first false
```

Binary Search on Real Values

- Generally for these problems, it’s recommended to use a constant number of binary search iterations. To be exact:

$$\log_2 \frac{max}{\varepsilon}$$

- https://codeforces.com/edu/course/2/lesson/6/2/practice/contest/283932/problem/E
- https://codeforces.com/edu/course/2/lesson/6/2/practice/contest/283932/problem/B

Resources

- Binary Search on real values:
  https://codeforces.com/blog/entry/63085
  https://codeforces.com/edu/course/2/lesson/6/2

Thanks for watching!