Binary Search

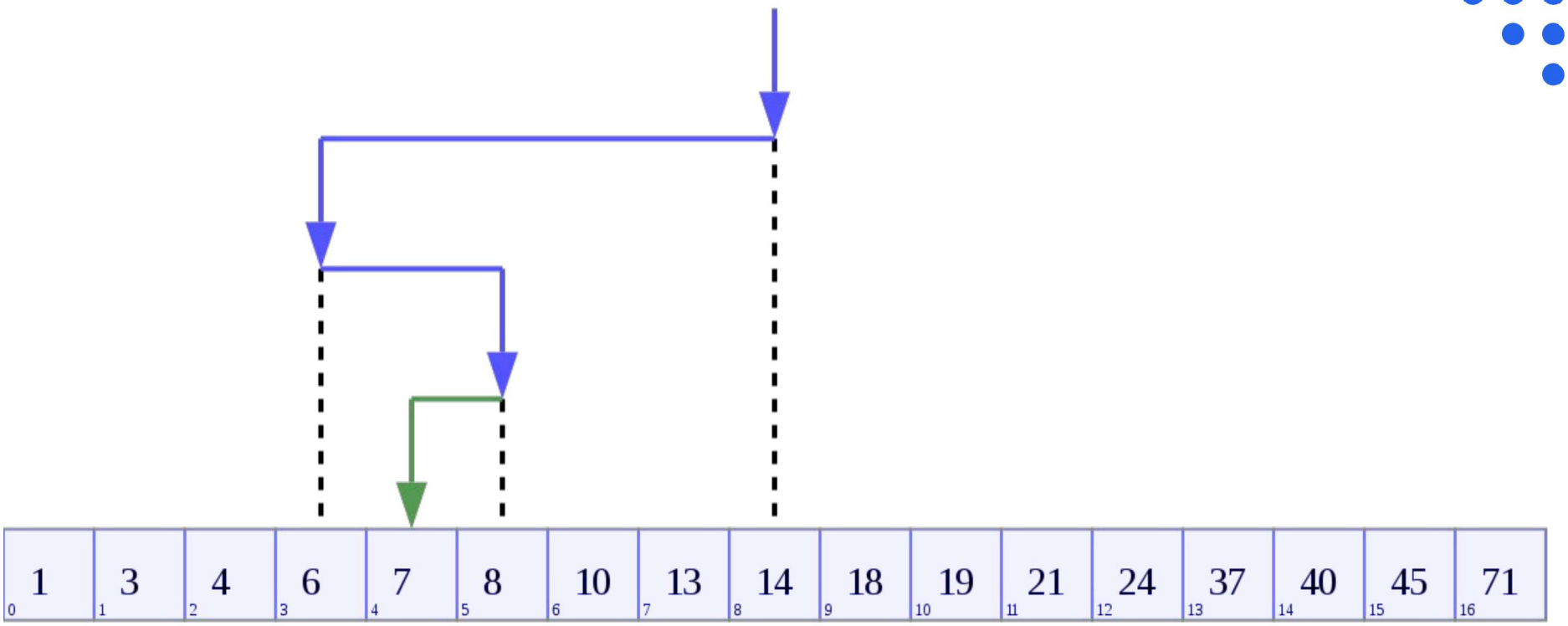
https://usaco.guide/silver/binary-search

CP Initiative

joincpi.org

Binary Search vs. Complete Search

- Binary Search finds an element in a **sorted** array in `O(log N)` time compared to complete search which takes `O(N)` time.
- Continuously splits up the array to locate the solution faster.
- In each iteration, it takes into consideration the middle element in the array.
  - If the value we are searching for is greater, we continue to search in the values above the middle element.
  - If the value is less, we continue to search in values less than the middle element.
- Interval becomes smaller and smaller until we locate the solution.


1 3 4 6 7 8 10 13 14 18 19 21 24 37 40 45 71
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Example Code

```
int[] arr = {1, 2, 5, 6, 7, 9, 11, 13};
int val = 9;
int left = 0, right = arr.length - 1;
while (left <= right) {
    int middle = (left + right) / 2;
    if (arr[middle] == val) {
        return middle;
    }
    if (arr[middle] < val) {
        left = middle + 1;
    } else {
        right = middle - 1;
    }
}
return -1;
```

Limitations

- Binary Search can only be applied on sorted values.
- Sorting takes `O(N log N)`
- Not limited to only arrays -- can be applied to any values that are monotonic.
  - For example, Binary Searching on a function.

Library functions

- Consult USACO Guide!
- There are typically built-in binary search functions for arrays.
- However, **know how to implement it yourself** since you may not always be binary searching arrays.

Example Code (Library)

```
int[] arr = {1, 2, 5, 6, 7, 9, 11, 13};
Arrays.sort(arr);
int val = 9;
int index = Arrays.binarySearch(arr, val);
return index;
```

Example Problem

[USACO - Counting Haybales (Silver)]

Counting Haybales Solution Sketch

- `0...10^9` is too large; we can't make an array of that length.
- Instead, let's put the locations of haybales in an array and sort it.
- Use binary search to count number of cows in a range `[A, B]` in `O(log N)` time:
  - First binary search to find the index of `B`
  - Then binary search to find the index of `A`
  - Then our answer is `IndexB - IndexA + 1`
- Try to implement with both library functions and with a custom implementation.

Counting Haybales Solution

[USACO Guide - Counting Haybales]

Binary Searching on Functions

CP Initiative
joincpi.org

Preconditions

- Binary search can only be used if the condition function is monotonic.
- A condition function `F` is monotonic if:
  - If `F(x)` is true, then `F(y)` must also be true for all `y <= x`
  - If `F(x)` is false, then `F(y)` must also be false for all `y >= x`
- If this is satisfied, then we can apply the extremely efficient Binary Search algorithm!

```
f(1) = true
f(2) = true
f(3) = true
f(4) = true
f(5) = true
f(6) = false
f(7) = false
f(8) = false
```

Implementation

- We want to construct a function `lastTrue` such that `lastTrue(lo, hi, f)` returns the last `x` in the range `[lo,hi]` such that `f(x) = true`.
- If no such `x` exists, then `lastTrue` should return `lo - 1`.
- (Similarly, `firstTrue` could return the first `x` such that `f(x) = true`.)

Implementation: Java

```java
public static int lastTrue(int lo, int hi, Predicate<Integer> f) {
    for (--lo; lo < hi; ) {
        int mid = lo+(hi-lo+1)/2;
        // find the middle of the current range (rounding up)
        if (f.test(mid)) lo = mid;
            // if mid works, then all numbers smaller than mid also work
        else hi = mid-1;
        // if mid does not work, greater values would not work too
        // so we don't care about them
    }
    return lo;
}
```

Implementation: C++

```
int lastTrue(int lo, int hi, function<bool(int)> f) {
    for (--lo; lo < hi; ) {
        int mid = lo+(hi-lo+1)/2;
        // find the middle of the current range (rounding up)
        if (f(mid)) lo = mid;
        // if mid works, then all numbers smaller than mid also work
        else hi = mid-1;
        // if mid does not work, greater values would not work too
        // so we don't care about them
    }
    return lo;
}
```

Example: Binary Search on Answer

CF 1201C - Maximum Median

Solution Sketch

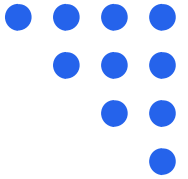
- Easier question: “Can I achieve a median of x?”
- Binary search on the answer (the maximum possible median).
- Check in $O(n)$ whether you can achieve each median.

Solution

[Maximum Median Solution - USACO Guide]

Challenge Problem 1: Angry Cows

[USACO Jan 2016 Silver - Angry Cows]



Challenge Problem 2: Magic Ship

[CF 1117C - Magic Ship]