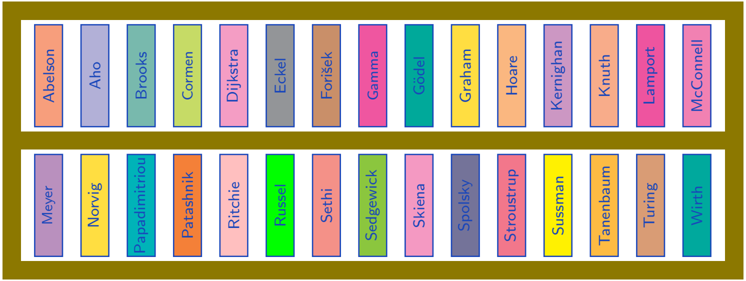
ITMO UNIVERSITY

How to Win Coding Competitions: Secrets of Champions

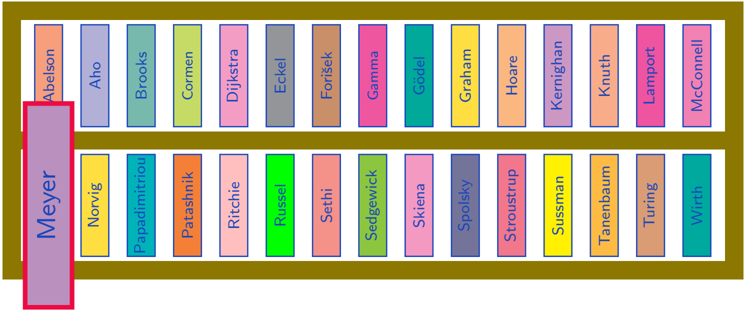
Week 3: Sorting and Search Algorithms
Lecture 10: Introduction to binary search

Maxim Buzdalov
Saint Petersburg 2016

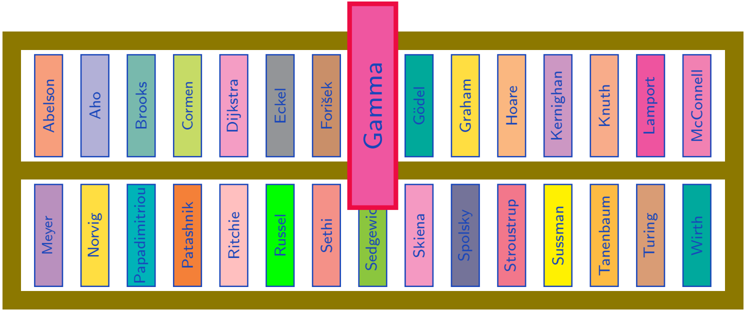
Recall the example from Lecture 1: how to search in sorted data



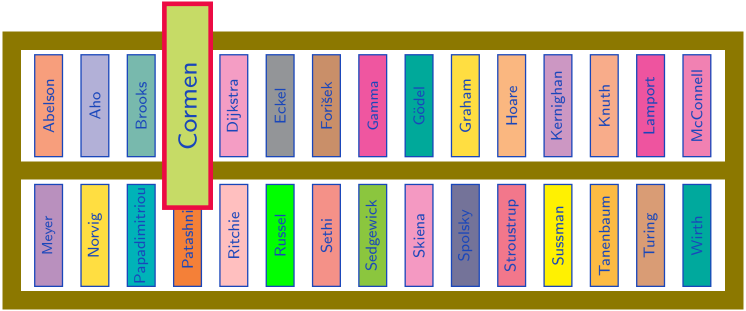
Recall the example from Lecture 1: how to search in sorted data



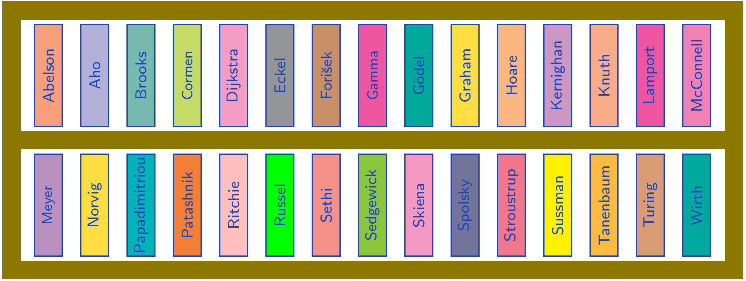
Recall the example from Lecture 1: how to search in sorted data



Recall the example from Lecture 1: how to search in sorted data



Recall the example from Lecture 1: how to search in sorted data



This is an example of binary search. We will have more in this video

A very general form of binary search

A very general form of binary search

- Given a function $F : D \to C$, such that:

A very general form of binary search

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an average-of-two operation

A very general form of binary search

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an average-of-two operation
    - We will write it $\textsc{Avg}(a, b)$ for arguments a and b
    - $\textsc{Avg}(a, b)$ should be between a and b and should not be equal to neither a nor b, unless there is no element of S between a and b

A very general form of binary search

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an average-of-two operation
    - We will write it $\textsc{Avg}(a, b)$ for arguments a and b
    - $\textsc{Avg}(a, b)$ should be between a and b and should not be equal to neither a nor b, unless there is no element of S between a and b
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\min} \le S, S \le D_{\max}\}$
    - Simply speaking, a piece of S between $D_{\min}$ and $D_{\max}$

A very general form of binary search

- Given a function $F : D \rightarrow C$, such that:
  - There exists a totally ordered set S equipped with an average-of-two operation
    - We will write it $\textsc{Avg}(a, b)$ for arguments a and b
    - $\textsc{Avg}(a, b)$ should be between a and b and should not be equal to neither a nor b, unless there is no element of S between a and b
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\text{min}} \le S, S \le D_{\text{max}}\}$
    - Simply speaking, a piece of S between D_{min} and D_{max}
  - $C = \{-1, 0, +1\}$ with the following meanings:
    - -1: "too early"
    - 0: "just in time"
    - $+1$: "too late"

A very general form of binary search

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an average-of-two operation
    - We will write it $\textsc{Avg}(a, b)$ for arguments a and b
    - $\textsc{Avg}(a, b)$ should be between a and b and should not be equal to neither a nor b, unless there is no element of S between a and b
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\min} \le S, S \le D_{\max}\}$
    - Simply speaking, a piece of S between $D_{\min}$ and $D_{\max}$
  - $C = \{-1, 0, +1\}$ with the following meanings:
    - -1: “too early”
    - 0: “just in time”
    - $+1$: “too late”
  - Monotonicity: Whenever $a < b$, $F(a) \le F(b)$

A very general form of binary search

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an average-of-two operation
    - We will write it $\textsc{Avg}(a, b)$ for arguments a and b
    - $\textsc{Avg}(a, b)$ should be between a and b and should not be equal to neither a nor b, unless there is no element of S between a and b
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\min} \le S, S \le D_{\max}\}$
    - Simply speaking, a piece of S between $D_{\min}$ and $D_{\max}$
  - $C = \{-1, 0, +1\}$ with the following meanings:
    - -1: "too early"
    - 0: "just in time"
    - $+1$: "too late"
  - Monotonicity: Whenever $a < b$, $F(a) \le F(b)$
- Need to find $x \in D$ such that $F(x) = 0$

A very general form of binary search

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an average-of-two operation
    - We will write it $\textsc{Avg}(a, b)$ for arguments a and b
    - $\textsc{Avg}(a, b)$ should be between a and b and should not be equal to neither a nor b, unless there is no element of S between a and b
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\text{min}} \le S, S \le D_{\text{max}}\}$
    - Simply speaking, a piece of S between D_{min} and D_{max}
  - $C = \{-1, 0, +1\}$ with the following meanings:
    - -1: "too early"
    - 0: "just in time"
    - $+1$: "too late"
  - Monotonicity: Whenever $a < b$, $F(a) \le F(b)$
- Need to find $x \in D$ such that $F(x) = 0$
  - Or, if impossible, find x and y as near as possible, such that $F(x) = -1$, $F(y) = 1$

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an $\textsc{Avg}(a, b)$ operation
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\text{min}} \le S, S \le D_{\text{max}}\}$
  - $C = \{-1\ (\text{"too early"}), 0\ (\text{"just in time"}), +1\ (\text{"too late"})\}$
  - Monotonicity: Whenever $a < b$, $F(a) \le F(b)$
- Need to find $x \in D$ such that $F(x) = 0$
  - Or, if impossible, find x, y as near as possible, such that $F(x) = -1$, $F(y) = 1$

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an $\textsc{Avg}(a, b)$ operation
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\text{min}} \le S, S \le D_{\text{max}}\}$
  - $C = \{-1\ (\text{“too early”}), 0\ (\text{”just in time”}), +1\ (\text{“too late”})\}$
  - Monotonicity: Whenever $a < b$, $F(a) \le F(b)$
- Need to find $x \in D$ such that $F(x) = 0$
  - Or, if impossible, find x, y as near as possible, such that $F(x) = -1$, $F(y) = 1$

Example: Find where q is in a sorted array

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an $\textsc{Avg}(a, b)$ operation
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\min} \le S, S \le D_{\max}\}$
  - $C = \{-1$ (“too early”), 0 (”just in time”), $+1$ (“too late”)$\}$
  - Monotonicity: Whenever $a < b$, $F(a) \le F(b)$
- Need to find $x \in D$ such that $F(x) = 0$
  - Or, if impossible, find x, y as near as possible, such that $F(x) = -1$, $F(y) = 1$

Example: Find where q is in a sorted array

- $D = [1; N]$: the set of array indices, ordered naturally
- $\textsc{Avg}(a, b) = \lfloor (a + b)/2 \rfloor$
- $F(x)$: “0” if $q = x$, “-1” if $x < q$, “$+1$” if $x > q$

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an $\textsc{Avg}(a, b)$ operation
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\min} \leq S, S \leq D_{\max}\}$
  - $C = \{-1$ ("too early"), 0 ("just in time"), $+1$ ("too late")$\}$
  - Monotonicity: Whenever $a < b$, $F(a) \leq F(b)$
- Need to find $x \in D$ such that $F(x) = 0$
  - Or, if impossible, find x, y as near as possible, such that $F(x) = -1$, $F(y) = 1$

Example: Find where q is in a sorted array

- $D = [1; N]$: the set of array indices, ordered naturally
- $\textsc{Avg}(a, b) = \lfloor (a + b)/2 \rfloor$
- $F(x)$: "0" if $q = x$, "-1" if $x < q$, "$+1$" if $x > q$
- If no such element, "0" is infeasible: then you will find where to insert q

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an $\textsc{Avg}(a, b)$ operation
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\text{min}} \leq S, S \leq D_{\text{max}}\}$
  - $C = \{-1 \text{ (``too early'')}, 0 \text{ (''just in time'')}, +1 \text{ (``too late'')}\}$
  - Monotonicity: Whenever $a < b$, $F(a) \leq F(b)$
- Need to find $x \in D$ such that $F(x) = 0$
  - Or, if impossible, find x, y as near as possible, such that $F(x) = -1$, $F(y) = 1$

Example: Find first occurence of q in a sorted array

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an $\textsc{Avg}(a, b)$ operation
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\min} \le S, S \le D_{\max}\}$
  - $C = \{-1$ (“too early”)$, 0$ (”just in time”)$, +1$ (“too late”)$\}$
  - Monotonicity: Whenever $a < b$, $F(a) \le F(b)$
- Need to find $x \in D$ such that $F(x) = 0$
  - Or, if impossible, find x, y as near as possible, such that $F(x) = -1$, $F(y) = 1$

Example: Find first occurence of q in a sorted array

- $D = [1; N]$: the set of array indices, ordered naturally
- $\textsc{Avg}(a, b) = \lfloor (a + b)/2 \rfloor$
- $F(x)$: “0” never, “-1” if $x < q$, “$+1$” if $x \ge q$

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an $\textsc{Avg}(a, b)$ operation
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\text{min}} \leq S, S \leq D_{\text{max}}\}$
  - $C = \{-1 \text{ (“too early”)}, 0 \text{ (”just in time”)}, +1 \text{ (“too late”)}\}$
  - Monotonicity: Whenever $a < b$, $F(a) \leq F(b)$
- Need to find $x \in D$ such that $F(x) = 0$
  - Or, if impossible, find x, y as near as possible, such that $F(x) = -1$, $F(y) = 1$

Example: Find first occurence of q in a sorted array

- $D = [1; N]$: the set of array indices, ordered naturally
- $\textsc{Avg}(a, b) = \lfloor (a + b)/2 \rfloor$
- $F(x)$: “0” never, “-1” if $x < q$, “$+1$” if $x \geq q$
- But first check if $F(y) = q\ldots$

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an $\textsc{Avg}(a, b)$ operation
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\text{min}} \le S, S \le D_{\text{max}}\}$
  - $C = \{-1\ (\text{“too early”}), 0\ (\text{”just in time”}), +1\ (\text{“too late”})\}$
  - Monotonicity: Whenever $a < b$, $F(a) \le F(b)$
- Need to find $x \in D$ such that $F(x) = 0$
  - Or, if impossible, find x, y as near as possible, such that $F(x) = -1$, $F(y) = 1$

Example: Find a root of a monotonically growing function f

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an $\textsc{Avg}(a, b)$ operation
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\min} \leq S, S \leq D_{\max}\}$
  - $C = \{-1 \text{ ("too early")}, 0 \text{ ("just in time")}, +1 \text{ ("too late")}\}$
  - Monotonicity: Whenever $a < b$, $F(a) \leq F(b)$
- Need to find $x \in D$ such that $F(x) = 0$
  - Or, if impossible, find x, y as near as possible, such that $F(x) = -1$, $F(y) = 1$

Example: Find a root of a monotonically growing function f

- $D = [\min; \max]$: the segment of $\mathbb{R}$ which we are interested in
- $\textsc{Avg}(a, b) = (a + b)/2$
- $F(x)$: "0" if $f(x) = 0$, "-1" if $f(x) < 0$, "$+1$" if $f(x) > 0$

- Given a function $F : D \to C$, such that:
  - There exists a totally ordered set S equipped with an $\textsc{Avg}(a, b)$ operation
  - D is a bounded subset of S: $D = \{s \mid s \in S, D_{\min} \le S, S \le D_{\max}\}$
  - $C = \{-1 \text{ (“too early”)}, 0 \text{ (”just in time”)}, +1 \text{ (“too late”)}\}$
  - Monotonicity: Whenever $a < b$, $F(a) \le F(b)$
- Need to find $x \in D$ such that $F(x) = 0$
  - Or, if impossible, find x, y as near as possible, such that $F(x) = -1$, $F(y) = 1$

Example: Find a root of a monotonically growing function f

- $D = [\min; \max]$: the segment of $\mathbb{R}$ which we are interested in
- $\textsc{Avg}(a, b) = (a + b)/2$
- $F(x)$: “0” if $f(x) = 0$, “-1” if $f(x) < 0$, “$+1$” if $f(x) > 0$
- Warning: you are unlikely to find the root exactly…

function $\textsc{BinarySearch}(F, \textsc{Avg}, D_{\min}, D_{\max})$
    $L \leftarrow D_{\min}$, $R \leftarrow D_{\max}$, $V_{\min} \leftarrow F(L)$, $V_{\max} \leftarrow F(R)$
    if $V_{\min} = 1$ **then return** $\langle \textsc{Null}, D_{\min} \rangle$ **end if**
    if $V_{\max} = -1$ **then return** $\langle D_{\max}, \textsc{Null} \rangle$ **end if**
    if $V_{\min} = 0$ **then return** $\langle D_{\min}, D_{\min} \rangle$ **end if**
    if $V_{\max} = 0$ **then return** $\langle D_{\max}, D_{\max} \rangle$ **end if**
    for ever do
        $M \leftarrow \textsc{Avg}(L, R)$
        if $M = L$ **or** $M = R$ **then return** $\langle L, R \rangle$ **end if**
        $v \leftarrow F(M)$
        if $v = 0$ **then return** $\langle M, M \rangle$ **end if**
        if $v = -1$ **then** $L \leftarrow M$ **else** $R \leftarrow M$ **end if**
    end for
end function

$$
\begin{array}{l}
\textbf{function } \textsc{BinarySearch}(F, \textsc{Avg}, D_{\min}, D_{\max}) \\
\quad L \leftarrow D_{\min},\ R \leftarrow D_{\max},\ V_{\min} \leftarrow F(L),\ V_{\max} \leftarrow F(R) \qquad \triangleright \text{ First evaluate endpoints} \\
\quad \textbf{if } V_{\min} = 1 \textbf{ then return } \langle \textsc{Null}, D_{\min} \rangle \textbf{ end if} \\
\quad \textbf{if } V_{\max} = -1 \textbf{ then return } \langle D_{\max}, \textsc{Null} \rangle \textbf{ end if} \\
\quad \textbf{if } V_{\min} = 0 \textbf{ then return } \langle D_{\min}, D_{\min} \rangle \textbf{ end if} \\
\quad \textbf{if } V_{\max} = 0 \textbf{ then return } \langle D_{\max}, D_{\max} \rangle \textbf{ end if} \\
\quad \textbf{for ever do} \\
\qquad M \leftarrow \textsc{Avg}(L, R) \\
\qquad \textbf{if } M = L \textbf{ or } M = R \textbf{ then return } \langle L, R \rangle \textbf{ end if} \\
\qquad v \leftarrow F(M) \\
\qquad \textbf{if } v = 0 \textbf{ then return } \langle M, M \rangle \textbf{ end if} \\
\qquad \textbf{if } v = -1 \textbf{ then } L \leftarrow M \textbf{ else } R \leftarrow M \textbf{ end if} \\
\quad \textbf{end for} \\
\textbf{end function}
\end{array}
$$

```
function BinarySearch(F, Avg, D_min, D_max)
    L ← D_min, R ← D_max, V_min ← F(L), V_max ← F(R)   ▷ First evaluate endpoints
    if V_min = 1 then return ⟨Null, D_min⟩ end if        ▷ If true, no zeros at all
    if V_max = −1 then return ⟨D_max, Null⟩ end if
    if V_min = 0 then return ⟨D_min, D_min⟩ end if
    if V_max = 0 then return ⟨D_max, D_max⟩ end if
    for ever do
        M ← Avg(L, R)
        if M = L or M = R then return ⟨L, R⟩ end if
        v ← F(M)
        if v = 0 then return ⟨M, M⟩ end if
        if v = −1 then L ← M else R ← M end if
    end for
end function
```

function BinarySearch(F, Avg, $D_{\min}$, $D_{\max}$)

  $L \leftarrow D_{\min}$, $R \leftarrow D_{\max}$, $V_{\min} \leftarrow F(L)$, $V_{\max} \leftarrow F(R)$ ▷ First evaluate endpoints

  if $V_{\min} = 1$ **then return** $\langle \text{Null}, D_{\min} \rangle$ **end if** ▷ If true, no zeros at all

  if $V_{\max} = -1$ **then return** $\langle D_{\max}, \text{Null} \rangle$ **end if** ▷ If true, no zeros at all

  if $V_{\min} = 0$ **then return** $\langle D_{\min}, D_{\min} \rangle$ **end if**

  if $V_{\max} = 0$ **then return** $\langle D_{\max}, D_{\max} \rangle$ **end if**

  for ever do

    $M \leftarrow \text{Avg}(L, R)$

    if $M = L$ **or** $M = R$ **then return** $\langle L, R \rangle$ **end if**

    $v \leftarrow F(M)$

    if $v = 0$ **then return** $\langle M, M \rangle$ **end if**

    if $v = -1$ **then** $L \leftarrow M$ **else** $R \leftarrow M$ **end if**

  end for

end function

function $\textsc{BinarySearch}(F, \textsc{Avg}, D_{\mathrm{min}}, D_{\mathrm{max}})$
  $L \leftarrow D_{\mathrm{min}},\ R \leftarrow D_{\mathrm{max}},\ V_{\mathrm{min}} \leftarrow F(L),\ V_{\mathrm{max}} \leftarrow F(R)$ ▷ First evaluate endpoints
  if $V_{\mathrm{min}} = 1$ **then return** $\langle \textsc{Null}, D_{\mathrm{min}} \rangle$ **end if** ▷ If true, no zeros at all
  if $V_{\mathrm{max}} = -1$ **then return** $\langle D_{\mathrm{max}}, \textsc{Null} \rangle$ **end if** ▷ If true, no zeros at all
  if $V_{\mathrm{min}} = 0$ **then return** $\langle D_{\mathrm{min}}, D_{\mathrm{min}} \rangle$ **end if** ▷ If true, zero is found
  if $V_{\mathrm{max}} = 0$ **then return** $\langle D_{\mathrm{max}}, D_{\mathrm{max}} \rangle$ **end if**
  for ever do
    $M \leftarrow \textsc{Avg}(L, R)$
    if $M = L$ **or** $M = R$ **then return** $\langle L, R \rangle$ **end if**
    $v \leftarrow F(M)$
    if $v = 0$ **then return** $\langle M, M \rangle$ **end if**
    if $v = -1$ **then** $L \leftarrow M$ **else** $R \leftarrow M$ **end if**
  end for
end function

function $\textsc{BinarySearch}(F, \textsc{Avg}, D_{\text{min}}, D_{\text{max}})$
    $L \leftarrow D_{\text{min}}, R \leftarrow D_{\text{max}}, V_{\text{min}} \leftarrow F(L), V_{\text{max}} \leftarrow F(R)$ ▷ First evaluate endpoints
    if $V_{\text{min}} = 1$ **then return** $\langle \textsc{Null}, D_{\text{min}} \rangle$ **end if** ▷ If true, no zeros at all
    if $V_{\text{max}} = -1$ **then return** $\langle D_{\text{max}}, \textsc{Null} \rangle$ **end if** ▷ If true, no zeros at all
    if $V_{\text{min}} = 0$ **then return** $\langle D_{\text{min}}, D_{\text{min}} \rangle$ **end if** ▷ If true, zero is found
    if $V_{\text{max}} = 0$ **then return** $\langle D_{\text{max}}, D_{\text{max}} \rangle$ **end if** ▷ If true, zero is found
    for ever do
        $M \leftarrow \textsc{Avg}(L, R)$
        if $M = L$ **or** $M = R$ **then return** $\langle L, R \rangle$ **end if**
        $v \leftarrow F(M)$
        if $v = 0$ **then return** $\langle M, M \rangle$ **end if**
        if $v = -1$ **then** $L \leftarrow M$ **else** $R \leftarrow M$ **end if**
    end for
end function

```
function BinarySearch(F, Avg, D_min, D_max)
    L ← D_min, R ← D_max, V_min ← F(L), V_max ← F(R)    ▷ First evaluate endpoints
    if V_min =  1 then return ⟨Null, D_min⟩ end if      ▷ If true, no zeros at all
    if V_max = −1 then return ⟨D_max, Null⟩ end if      ▷ If true, no zeros at all
    if V_min =  0 then return ⟨D_min, D_min⟩ end if     ▷ If true, zero is found
    if V_max =  0 then return ⟨D_max, D_max⟩ end if     ▷ If true, zero is found
    for ever do                                         ▷ Invariant: F(L) = −1, F(R) = 1
        M ← Avg(L, R)
        if M = L or M = R then return ⟨L, R⟩ end if
        v ← F(M)
        if v = 0 then return ⟨M, M⟩ end if
        if v = −1 then L ← M else R ← M end if
    end for
end function
```

function $\textsc{BinarySearch}(F, \textsc{Avg}, D_{\min}, D_{\max})$
  $L \leftarrow D_{\min}$, $R \leftarrow D_{\max}$, $V_{\min} \leftarrow F(L)$, $V_{\max} \leftarrow F(R)$ ▷ First evaluate endpoints
  if $V_{\min} = 1$ **then return** $\langle \textsc{Null}, D_{\min} \rangle$ **end if** ▷ If true, no zeros at all
  if $V_{\max} = -1$ **then return** $\langle D_{\max}, \textsc{Null} \rangle$ **end if** ▷ If true, no zeros at all
  if $V_{\min} = 0$ **then return** $\langle D_{\min}, D_{\min} \rangle$ **end if** ▷ If true, zero is found
  if $V_{\max} = 0$ **then return** $\langle D_{\max}, D_{\max} \rangle$ **end if** ▷ If true, zero is found
  for ever do ▷ Invariant: $F(L) = -1$, $F(R) = 1$
    $M \leftarrow \textsc{Avg}(L, R)$ ▷ Getting new query point
    if $M = L$ **or** $M = R$ **then return** $\langle L, R \rangle$ **end if**
    $v \leftarrow F(M)$
    if $v = 0$ **then return** $\langle M, M \rangle$ **end if**
    if $v = -1$ **then** $L \leftarrow M$ **else** $R \leftarrow M$ **end if**
  end for
end function

```
function BinarySearch(F, Avg, D_min, D_max)
    L ← D_min, R ← D_max, V_min ← F(L), V_max ← F(R)      ▷ First evaluate endpoints
    if V_min =  1 then return ⟨Null, D_min⟩ end if          ▷ If true, no zeros at all
    if V_max = −1 then return ⟨D_max, Null⟩ end if          ▷ If true, no zeros at all
    if V_min =  0 then return ⟨D_min, D_min⟩ end if         ▷ If true, zero is found
    if V_max =  0 then return ⟨D_max, D_max⟩ end if         ▷ If true, zero is found
    for ever do                                              ▷ Invariant: F(L) = −1, F(R) = 1
        M ← Avg(L, R)                                        ▷ Getting new query point
        if M = L or M = R then return ⟨L, R⟩ end if          ▷ (L, R) empty → no zeros
        v ← F(M)
        if v = 0 then return ⟨M, M⟩ end if
        if v = −1 then L ← M else R ← M end if
    end for
end function
```

function $\textsc{BinarySearch}(F, \textsc{Avg}, D_{\min}, D_{\max})$
- $L \leftarrow D_{\min},\ R \leftarrow D_{\max},\ V_{\min} \leftarrow F(L),\ V_{\max} \leftarrow F(R)$ ▷ First evaluate endpoints
- **if** $V_{\min} = 1$ **then return** $\langle \textsc{Null}, D_{\min} \rangle$ **end if** ▷ If true, no zeros at all
- **if** $V_{\max} = -1$ **then return** $\langle D_{\max}, \textsc{Null} \rangle$ **end if** ▷ If true, no zeros at all
- **if** $V_{\min} = 0$ **then return** $\langle D_{\min}, D_{\min} \rangle$ **end if** ▷ If true, zero is found
- **if** $V_{\max} = 0$ **then return** $\langle D_{\max}, D_{\max} \rangle$ **end if** ▷ If true, zero is found
- **for ever do** ▷ Invariant: $F(L) = -1,\ F(R) = 1$
  - $M \leftarrow \textsc{Avg}(L, R)$ ▷ Getting new query point
  - **if** $M = L$ **or** $M = R$ **then return** $\langle L, R \rangle$ **end if** ▷ (L, R) empty $\rightarrow$ no zeros
  - $v \leftarrow F(M)$ ▷ Evaluating M
  - **if** $v = 0$ **then return** $\langle M, M \rangle$ **end if**
  - **if** $v = -1$ **then** $L \leftarrow M$ **else** $R \leftarrow M$ **end if**
- **end for**

end function

```
function BinarySearch(F, Avg, D_min, D_max)
    L ← D_min, R ← D_max, V_min ← F(L), V_max ← F(R)        ▷ First evaluate endpoints
    if V_min =  1 then return ⟨Null, D_min⟩ end if          ▷ If true, no zeros at all
    if V_max = −1 then return ⟨D_max, Null⟩ end if          ▷ If true, no zeros at all
    if V_min =  0 then return ⟨D_min, D_min⟩ end if         ▷ If true, zero is found
    if V_max =  0 then return ⟨D_max, D_max⟩ end if         ▷ If true, zero is found
    for ever do                                 ▷ Invariant: F(L) = −1, F(R) = 1
        M ← Avg(L, R)                                       ▷ Getting new query point
        if M = L or M = R then return ⟨L, R⟩ end if         ▷ (L, R) empty → no zeros
        v ← F(M)                                            ▷ Evaluating M
        if v = 0 then return ⟨M, M⟩ end if                  ▷ Direct hit!
        if v = −1 then L ← M else R ← M end if
    end for
end function
```

```
function BinarySearch(F, Avg, D_min, D_max)
    L ← D_min, R ← D_max, V_min ← F(L), V_max ← F(R)    ▷ First evaluate endpoints
    if V_min = 1 then return ⟨Null, D_min⟩ end if         ▷ If true, no zeros at all
    if V_max = −1 then return ⟨D_max, Null⟩ end if        ▷ If true, no zeros at all
    if V_min = 0 then return ⟨D_min, D_min⟩ end if        ▷ If true, zero is found
    if V_max = 0 then return ⟨D_max, D_max⟩ end if        ▷ If true, zero is found
    for ever do                                  ▷ Invariant: F(L) = −1, F(R) = 1
        M ← Avg(L, R)                                     ▷ Getting new query point
        if M = L or M = R then return ⟨L, R⟩ end if       ▷ (L, R) empty → no zeros
        v ← F(M)                                          ▷ Evaluating M
        if v = 0 then return ⟨M, M⟩ end if                ▷ Direct hit!
        if v = −1 then L ← M else R ← M end if            ▷ "Too early": use right part
    end for
end function
```

function $\textsc{BinarySearch}(F, \textsc{Avg}, D_{\text{min}}, D_{\text{max}})$
 $L \leftarrow D_{\text{min}}, R \leftarrow D_{\text{max}}, V_{\text{min}} \leftarrow F(L), V_{\text{max}} \leftarrow F(R)$ ▷ First evaluate endpoints
 if $V_{\text{min}} = 1$ **then return** $\langle \textsc{Null}, D_{\text{min}} \rangle$ **end if** ▷ If true, no zeros at all
 if $V_{\text{max}} = -1$ **then return** $\langle D_{\text{max}}, \textsc{Null} \rangle$ **end if** ▷ If true, no zeros at all
 if $V_{\text{min}} = 0$ **then return** $\langle D_{\text{min}}, D_{\text{min}} \rangle$ **end if** ▷ If true, zero is found
 if $V_{\text{max}} = 0$ **then return** $\langle D_{\text{max}}, D_{\text{max}} \rangle$ **end if** ▷ If true, zero is found
 for ever do ▷ Invariant: $F(L) = -1, F(R) = 1$
  $M \leftarrow \textsc{Avg}(L, R)$ ▷ Getting new query point
  if $M = L$ **or** $M = R$ **then return** $\langle L, R \rangle$ **end if** ▷ (L, R) empty $\rightarrow$ no zeros
  $v \leftarrow F(M)$ ▷ Evaluating M
  if $v = 0$ **then return** $\langle M, M \rangle$ **end if** ▷ Direct hit!
  if $v = -1$ **then** $L \leftarrow M$ **else** $R \leftarrow M$ **end if** ▷ "Too early": use right part
 end for ▷ "Too late": use left part
end function

Termination

Termination

- Guaranteed to terminate if D is finite

Termination

- Guaranteed to terminate if D is finite
- Proof: $[L, R]$ shrinks at least by one item on every iteration

Correctness

Termination

- Guaranteed to terminate if D is finite
- Proof: $[L, R]$ shrinks at least by one item on every iteration

Correctness

- Loop invariant: $F(L) = -1$, $F(R) = 1$

Termination

- Guaranteed to terminate if D is finite
- Proof: $[L, R]$ shrinks at least by one item on every iteration

Correctness

- Loop invariant: $F(L) = -1$, $F(R) = 1$
- Zeros are always between $\rightarrow$ will be found if exist

Termination

- Guaranteed to terminate if D is finite
- Proof: $[L, R]$ shrinks at least by one item on every iteration

Correctness

- Loop invariant: $F(L) = -1$, $F(R) = 1$
- Zeros are always between $\rightarrow$ will be found if exist
- Nonexisting zeros: will report a point where -1 switches to 1

Running time

Termination

- Guaranteed to terminate if D is finite
- Proof: $[L, R]$ shrinks at least by one item on every iteration

Correctness

- Loop invariant: $F(L) = -1$, $F(R) = 1$
- Zeros are always between $\rightarrow$ will be found if exist
- Nonexisting zeros: will report a point where -1 switches to 1

Running time

- Strongly depends on properties of $\textsc{Avg}$

Termination

- Guaranteed to terminate if D is finite
- Proof: $[L, R]$ shrinks at least by one item on every iteration

Correctness

- Loop invariant: $F(L) = -1$, $F(R) = 1$
- Zeros are always between $\rightarrow$ will be found if exist
- Nonexisting zeros: will report a point where -1 switches to 1

Running time

- Strongly depends on properties of $\textsc{Avg}$
- If $\textsc{Avg}$ is a "real" average, the running time is $O(\log D)$
  - the size of $[L; R)$ range is divided by two on every iteration