Rökstudd forritun
Reasoned Programming

Helmingunarleit – Binary Search

Snorri Agnarsson
snorri@hi.is

Helmingunarleit (English version is in he second half of the slides)

- Mjög hraðvirk leitaraðferð
- Flóknari en línuleg leit
- Krefst þess að gildin séu þegar röðuð
- Afar mikilvæg og algeng aðferð
- Allir þurfa að kunna helmingunarleit

Helmingunarleit, dæmi

- Leitum að 19 í röðuðu rununni 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22

Helmingunarleit, dæmi

- Leitum að 19 í röðuðu rununni 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Í upphafi er óþekkta svæðið öll runan og svæðin < 19 og ≥ 19 eru tóm

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Helmingunarleit, dæmi

- Leitum að 19 í röðuðu rununni $1\ 2\ 3\ 5\ 6\ 7\ 8\ 10\ 12\ 13\ 15\ 16\ 18\ 19\ 20\ 22$
- Í upphafi er óþekkta svæðið öll runan og svæðin < 19 og ≥ 19 eru tóm

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Runan inniheldur 16 gildi og miðjusætið er því sæti 8, sem inniheldur $10 < 19$

Helmingunarleit, dæmi

- Leitum að 19 í röðuðu rununni $1\ 2\ 3\ 5\ 6\ 7\ 8\ 10\ 12\ 13\ 15\ 16\ 18\ 19\ 20\ 22$
- Í upphafi er óþekkta svæðið öll runan og svæðin < 19 og ≥ 19 eru tóm

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Runan inniheldur 16 gildi og miðjusætið er því sæti 8, sem inniheldur $10 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Helmingunarleit, dæmi

- Leitum að 19 í röðuðu rununni 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Í upphafi er óþekkta svæðið öll runan og svæðin < 19 og ≥ 19 eru tóm

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Runan inniheldur 16 gildi og miðjusætið er því sæti 8, sem inniheldur $10 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Næsta miðjusæti er sæti 12 sem inniheldur $16 < 19$

Helmingunarleit, dæmi

- Leitum að 19 í röðuðu rununni 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Í upphafi er óþekkta svæðið öll runan og svæðin < 19 og ≥ 19 eru tóm

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Runan inniheldur 16 gildi og miðjusætið er því sæti 8, sem inniheldur $10 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Næsta miðjusæti er sæti 12 sem inniheldur $16 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Helmingunarleit, dæmi

- Leitum að 19 í röðuðu rununni 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Í upphafi er óþekkta svæðið öll runan og svæðin < 19 og ≥ 19 eru tóm

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Runan inniheldur 16 gildi og miðjusætið er því sæti 8, sem inniheldur $10 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Næsta miðjusæti er sæti 12 sem inniheldur $16 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Næsta miðjusæti er sæti 14 sem inniheldur $19 \geq 19$

Helmingunarleit, dæmi

- Leitum að 19 í röðuðu rununni 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Í upphafi er óþekkta svæðið öll runan og svæðin < 19 og ≥ 19 eru tóm

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Runan inniheldur 16 gildi og miðjusætið er því sæti 8, sem inniheldur $10 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Næsta miðjusæti er sæti 12 sem inniheldur $16 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Næsta miðjusæti er sæti 14 sem inniheldur $19 \geq 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Helmingunarleit, dæmi

- Leitum að 19 í röðuðu rununni 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Í upphafi er óþekkta svæðið öll runan og svæðin < 19 og ≥ 19 eru tóm

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Runan inniheldur 16 gildi og miðjusætið er því sæti 8, sem inniheldur $10 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Næsta miðjusæti er sæti 12 sem inniheldur $16 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Næsta miðjusæti er sæti 14 sem inniheldur $19 \geq 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Næsta miðjusæti er sæti 13 sem inniheldur $18 < 19$

Helmingunarleit, dæmi

- Leitum að 19 í röðuðu rununni 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Í upphafi er óþekkta svæðið öll runan og svæðin < 19 og ≥ 19 eru tóm

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Runan inniheldur 16 gildi og miðjusætið er því sæti 8, sem inniheldur $10 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Næsta miðjusæti er sæti 12 sem inniheldur $16 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Næsta miðjusæti er sæti 14 sem inniheldur $19 \geq 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Næsta miðjusæti er sæti 13 sem inniheldur $18 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- Óþekkta svæðið er tómt og leitinni er lokið

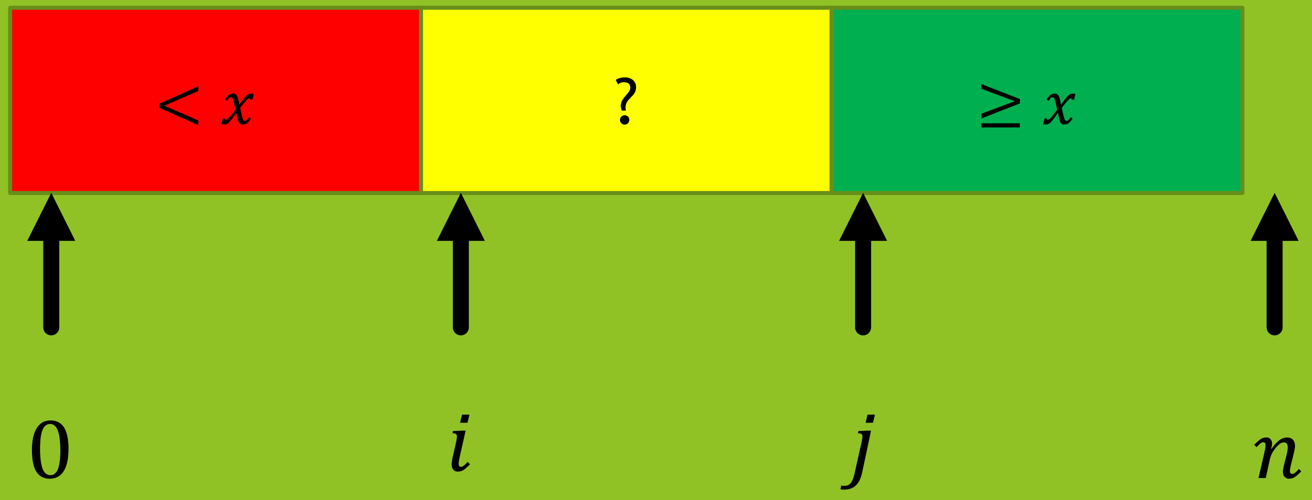
Grunnhugmynd helmingunarleitar

Fastayrðing lykkju:

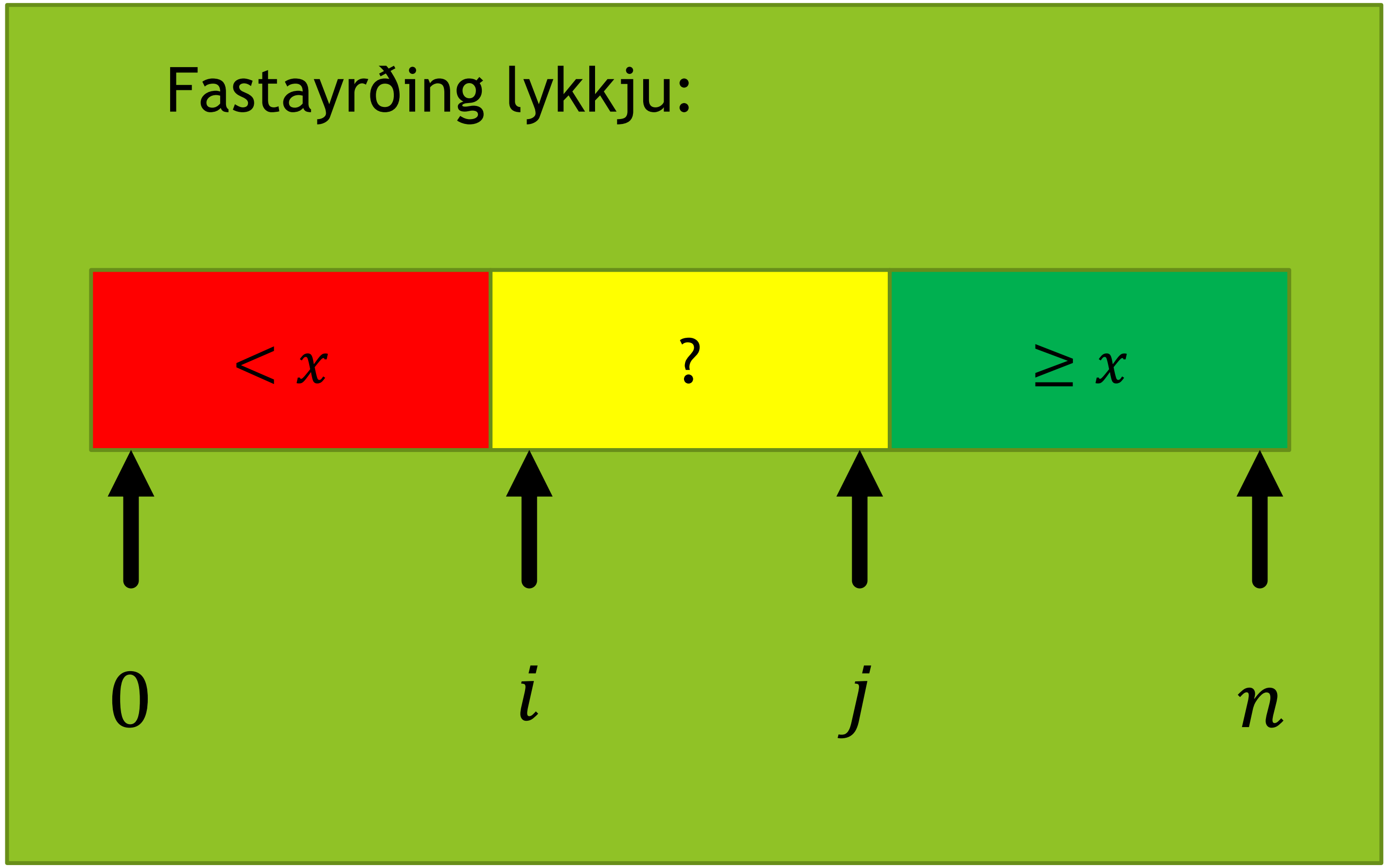
$< x$	?	$\geq x$

Nákvæm útfærsla

Fastayrðing lykkju:



Ýmsir aðrir möguleikar



Helmingunarleit (binary search)

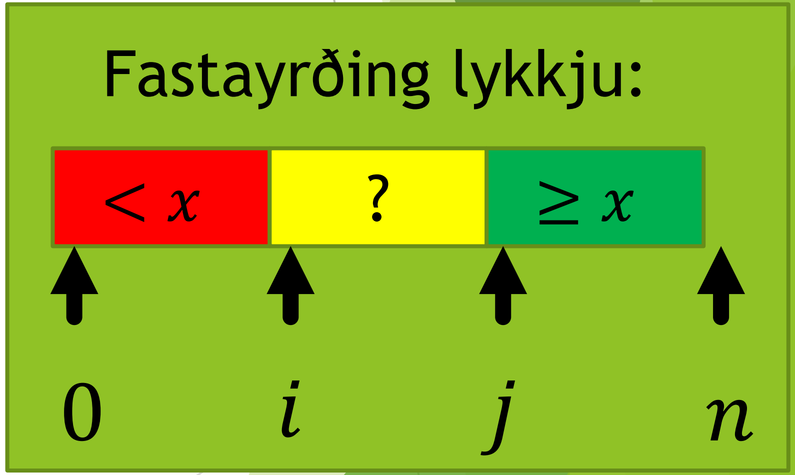
{
Notkun: i := leita($x, a_0, a_1, \ldots, a_{n-1}$)
Fyrir: x er heiltala,
$a_0, a_1, \ldots, a_{n-1}$ eru heiltölur í vaxandi röð
Eftir: $0 \leq i \leq n, \quad a_0, \ldots, a_{i-1} < x \leq a_i, \ldots, a_{n-1}$
}

stef leita(x : heiltala, $a_0, a_1, \ldots, a_{n-1}$: heiltölur)
    ???

Helmingunarleit (binary search)

{
Notkun: i := leita($x, a_0, a_1, \dots, a_{n-1}$)
Fyrir: x er heiltala,
$a_0, a_1, \dots, a_{n-1}$ eru heiltölur í vaxandi röð
Eftir: $0 \le i \le n$, $a_0, \dots, a_{i-1} < x \le a_i, \dots, a_{n-1}$
}

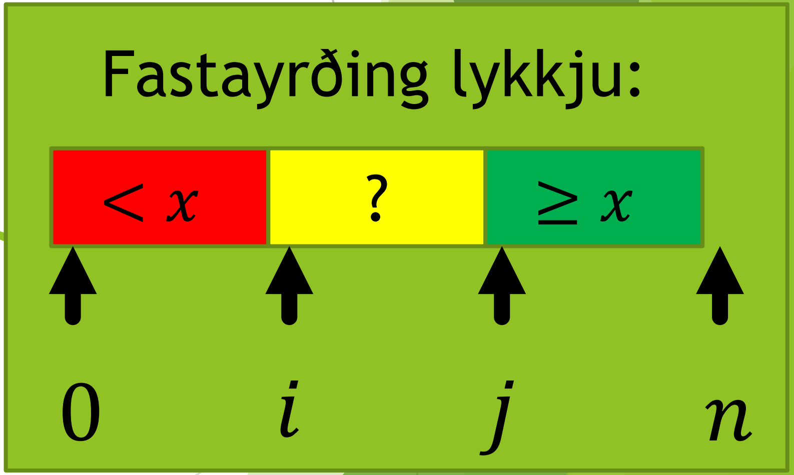
stef leita(x : heiltala, $a_1, a_2, \dots, a_{n-1}$: heiltölur)
    ???



Helmingunarleit (binary search)

{
Notkun: $i := \mathrm{leita}(x, a_0, a_1, \dots, a_{n-1})$
Fyrir: x er heiltala,
$a_0, a_1, \dots, a_{n-1}$ eru heiltölur í vaxandi röð
Eftir: $0 \leq i \leq n$, $a_0, \dots, a_{i-1} < x \leq a_i, \dots, a_{n-1}$
}

stef leita(x : heiltala, $a_1, a_2, \dots, a_{n-1}$: heiltölur)
    ???
    meðan ???
        { $0 \leq i \leq j \leq n$, $a_0, \dots, a_{i-1} < x \leq a_j, \dots, a_{n-1}$ }
        ???
    ???

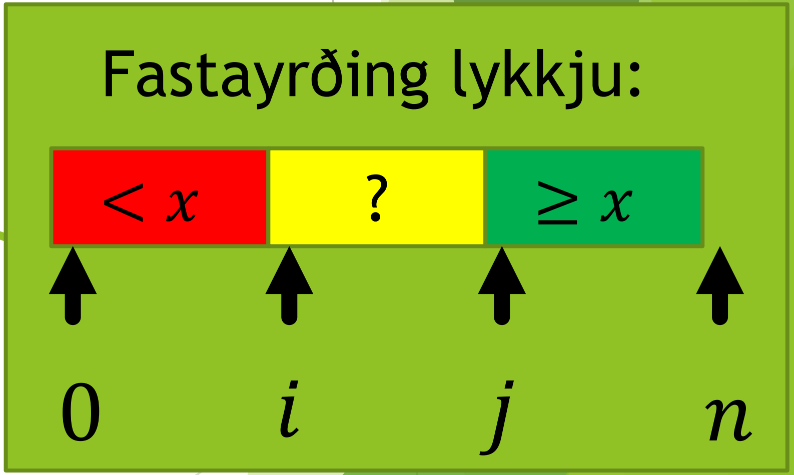


Helmingunarleit (binary search)

{
Notkun: i := leita(x, $a_0, a_1, \ldots, a_{n-1}$)
Fyrir: x er heiltala,
$a_0, a_1, \ldots, a_{n-1}$ eru heiltölur í vaxandi röð
Eftir: $0 \le i \le n$, $a_0, \ldots, a_{i-1} < x \le a_i, \ldots, a_{n-1}$
}

stef leita(x : heiltala, $a_1, a_2, \ldots, a_{n-1}$: heiltölur)

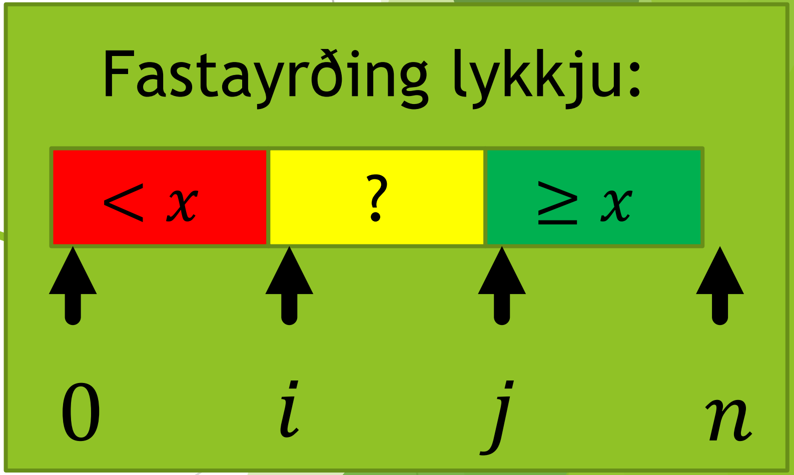
    ???
    meðan $i \neq j$
        { $0 \le i \le j \le n$, $a_0, \ldots, a_{i-1} < x \le a_j, \ldots, a_{n-1}$ }
        ???
    ???



Helmingunarleit (binary search)

{
Notkun: $i := \text{leita}(x, a_0, a_1, \dots, a_{n-1})$
Fyrir: x er heiltala,
$a_0, a_1, \dots, a_{n-1}$ eru heiltölur í vaxandi röð
Eftir: $0 \le i \le n$, $a_0, \dots, a_{i-1} < x \le a_i, \dots, a_{n-1}$
}

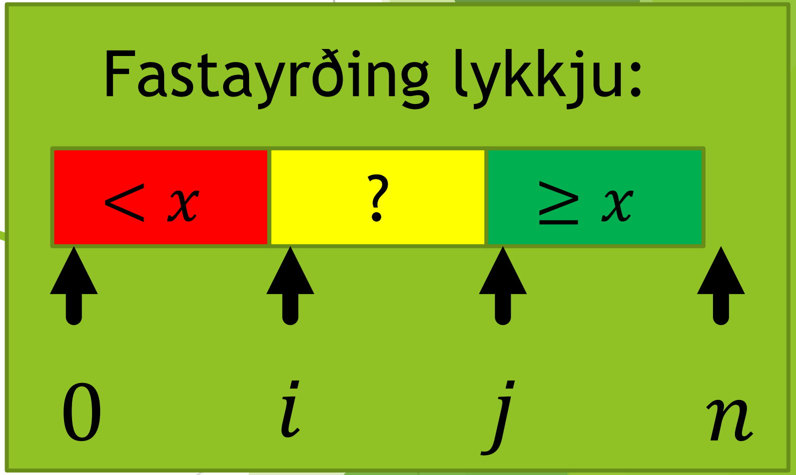
stef leita(x : heiltala, $a_1, a_2, \dots, a_{n-1}$: heiltölur)
   ???
   meðan $i \ne j$
      { $0 \le i \le j \le n$, $a_0, \dots, a_{i-1} < x \le a_j, \dots, a_{n-1}$ }
      ???
   skila i



Helmingunarleit (binary search)

{
Notkun: i := leita($x, a_0, a_1, \dots, a_{n-1}$)
Fyrir: x er heiltala,
$a_0, a_1, \dots, a_{n-1}$ eru heiltölur í vaxandi röð
Eftir: $0 \le i \le n$, $a_0, \dots, a_{i-1} < x \le a_i, \dots, a_{n-1}$
}

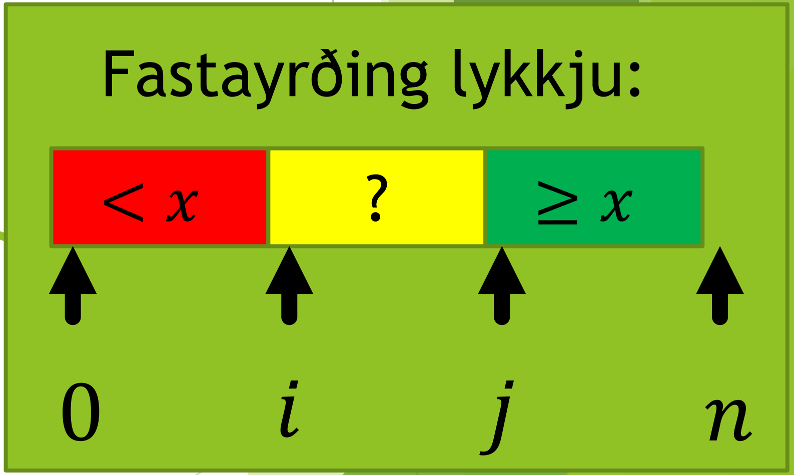
stef leita(x : heiltala, $a_1, a_2, \dots, a_{n-1}$: heiltölur)
  i := 0; j := n
  meðan $i \ne j$
    { $0 \le i \le j \le n$, $a_0, \dots, a_{i-1} < x \le a_j, \dots, a_{n-1}$ }
    ???
  skila i



Helmingunarleit (binary search)

{
Notkun: i := leita(x, $a_0, a_1, \ldots, a_{n-1}$)
Fyrir: x er heiltala,
$a_0, a_1, \ldots, a_{n-1}$ eru heiltölur í vaxandi röð
Eftir: $0 \leq i \leq n$, $a_0, \ldots, a_{i-1} < x \leq a_i, \ldots, a_{n-1}$
}

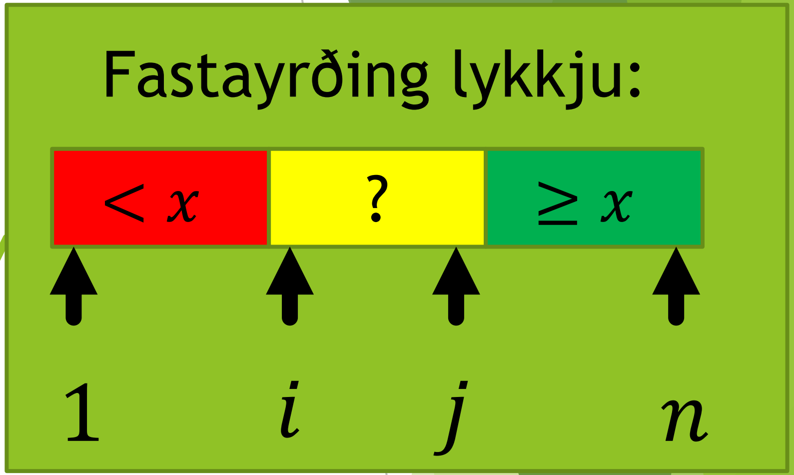
stef leita(x : heiltala, $a_1, a_2, \ldots, a_{n-1}$: heiltölur)
    i := 0; j := n
    meðan $i \neq j$
        { $0 \leq i \leq j \leq n$, $a_0, \ldots, a_{i-1} < x \leq a_j, \ldots, a_{n-1}$ }
        m := $\lfloor (i+j)/2 \rfloor$
        ef $a_m < x$ **þá** i := $m + 1$
        annars j := m
    skila i



Sama og $i + \lfloor (j-i)/2 \rfloor$

Helmingunarleit (önnur útgáfa)

{
Notkun: i := leita(x, $a_1, a_2, \dots, a_n$)
Fyrir: x er heiltala,
$a_1, a_2, \dots, a_n$ eru heiltölur í vaxandi röð
Eftir: $1 \le i \le n+1$, $a_1, \dots, a_{i-1} < x \le a_i, \dots, a_n$
}



stef leita(x : heiltala, $a_1, a_2, \dots, a_n$: heiltölur)
    i := 1; j := n
    meðan $i \le j$
        { $1 \le i \le j+1 \le n+1$, $a_1, \dots, a_{i-1} < x \le a_{j+1}, \dots, a_n$ }
        m := $\lfloor (i+j)/2 \rfloor$
        ef $a_m < x$ **þá** i := $m+1$
        annars j := $m-1$
    skila i

Helmingunarleit (enn önnur útgáfa)

{

Notkun: i := leita(x, $a_0, a_1, \dots, a_{n-1}$)

Fyrir: x er heiltala,

$a_0, a_1, \dots, a_{n-1}$ eru heiltölur í vaxandi röð

Eftir: $0 \leq i < n$ og $a_i = x$ EÐA

$i < 0$ og $a_0, a_1, \dots, a_{n-1} \neq x$

}

stef leita(x : heiltala, $a_1, a_2, \dots, a_{n-1}$: heiltölur)

    i := 0; j := n

    meðan $i \neq j$

        { $0 \leq i \leq j \leq n$, $a_0, \dots, a_{i-1} < x < a_j, \dots, a_{n-1}$ }

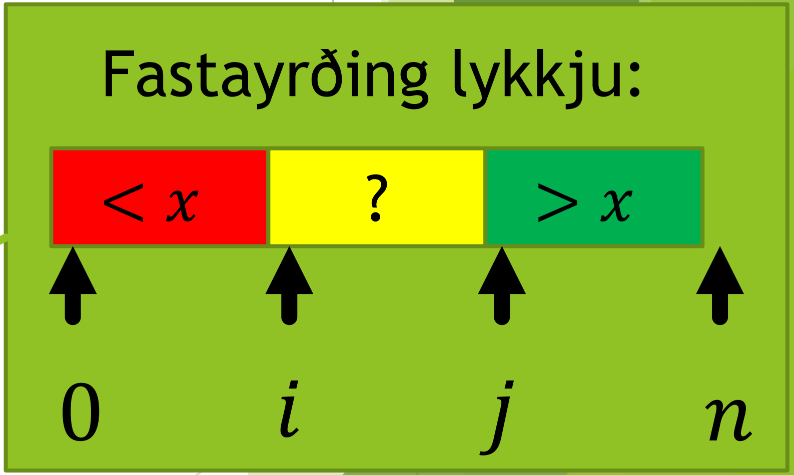
        m := $\lfloor (i+j)/2 \rfloor$

        ef $a_m = x$ **þá**   **skila** i

        ef $a_m < x$ **þá** i := $m + 1$

        annars j := m

    skila -1



Helmingunarleit (enn önnur útgáfa)

{

Notkun: i := leita(x, $a_0, a_1, \dots, a_{n-1}$)

Fyrir: x er heiltala,

$a_0, a_1, \dots, a_{n-1}$ eru heiltölur í vaxandi röð

Eftir: $0 \leq i < n$ og $a_i = x$ EÐA

$i < 0$ og $a_0, a_1, \dots, a_{n-1} \neq x$

}

stef leita(x : heiltala, $a_1, a_2, \dots, a_{n-1}$: heiltölur)

    i := 0; j := n

    meðan $i \neq j$

        { $0 \leq i \leq j \leq n$, $a_0, \dots, a_{i-1} < x < a_j, \dots, a_{n-1}$ }

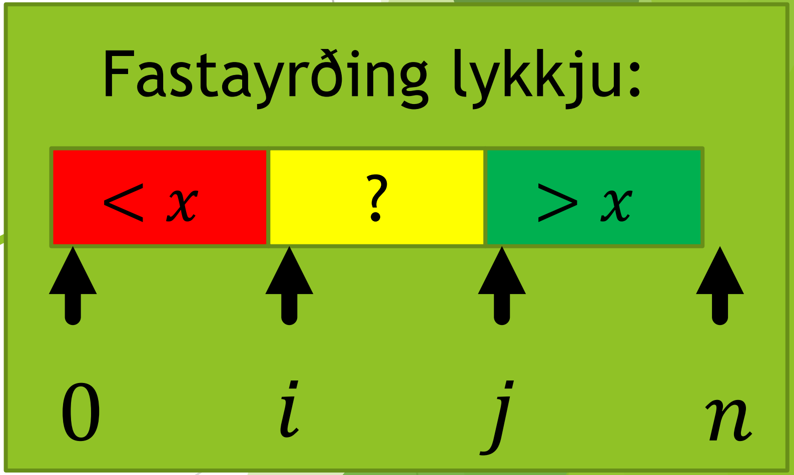
        m := $\lfloor (i+j)/2 \rfloor$

        ef $a_m = x$ **þá**   **skila** i

        ef $a_m < x$ **þá** i := $m + 1$

        annars j := m

    skila $-\mathrm{i} - 1$



Hvers vegna er þetta sniðugt skilagildi?

Helmingunarleit (enn önnur útgáfa)

{
Notkun: i := leita($x, a_0, a_1, \dots, a_{n-1}$)
Fyrir: x er heiltala,
$a_0, a_1, \dots, a_{n-1}$ eru heiltölur í vaxandi röð
Eftir: $0 \le i < n$ og $a_i = x$ EÐA
$0 \le -i-1 \le n$ og $a_0, a_1, \dots, a_{-i-2} < x < a_{-i-1}, \dots, a_{n-1}$
}

stef leita(x : heiltala, $a_1, a_2, \dots, a_{n-1}$: heiltölur)
    i := 0; j := n
    meðan $i \ne j$
        { $0 \le i \le j \le n$, $a_0, \dots, a_{i-1} < x < a_j, \dots, a_{n-1}$ }
        m := $\lfloor (i+j)/2 \rfloor$
        ef $a_m = x$ **þá**   **skila** i
        ef $a_m < x$ **þá** i := $m + 1$
        annars j := m
    skila $-i-1$

Fastayrðing lykkju:

$< x$	?	$> x$

0 i j n

Hvers vegna er þetta sniðugt skilagildi?

Helmingunarleit í Dafny

```
method Search( a: array<int>, i: int, j: int, x: int ) returns( k: int )
    requires 0 <= i <= j <= a.Length;
    requires forall p,q | i <= p < q < j :: a[p] <= a[q];
    ensures i <= k <= j;
    ensures forall r | i <= r < k :: a[r] < x;
    ensures forall r | k <= r < j :: a[r] >= x;
{
    var p,q := i,j;
    while p != q
        invariant i <= p <= q <= j;
        decreases q-p;
        invariant forall r | i <= r < p :: a[r] < x;
        invariant forall r | q <= r < j :: a[r] >= x;
    {
        var m := p+(q-p)/2;
        if a[m] < x { p := m+1; }
        else        { q := m;   }
    }
    k := p;
}
```

Sjá einnig Search.dfy
í Canvas

Helmingunarleit í Dafny

```
method Search( a: array<int>, i: int, j: int, x: int ) returns( k: int )
    decreases j-i;
    requires 0 <= i <= j <= a.Length;
    requires forall p,q | i <= p < q < j :: a[p] <= a[q];
    ensures k < 0 ==> !(x in a[i..j]);
    ensures k < 0 ==> i <= -k-1 <= j;
    ensures k < 0 ==> forall r | i <= r < -k-1 :: a[r] < x;
    ensures k < 0 ==> forall r | -k-1 <= r < j :: a[r] > x;
    ensures k >= 0 ==> i <= k < j;
    ensures k >= 0 ==> a[k] == x;
{
    if i == j { return -1-i; }
    var m := i+(j-i)/2;
    if a[m] == x { return m; }
    if a[m] < x { k := Search(a,m+1,j,x); }
    else        { k := Search(a,i,m,x);   }
}
```

Helmingunarleit í Dafny

```
method Search( a: array<int>, i: int, j: int, x: int ) returns( k: int )
    requires 0 <= i <= j <= a.Length;
    requires forall p,q | i <= p < q < j :: a[p] <= a[q];
    ensures k < 0 ==> !(x in a[i..j]);
    ensures k < 0 ==> i <= -k-1 <= j;
    ensures k < 0 ==> forall r | i <= r < -k-1 :: a[r] < x;
    ensures k < 0 ==> forall r | -k-1 <= r < j :: a[r] > x;
    ensures k >= 0 ==> i <= k < j;
    ensures k >= 0 ==> a[k] == x;
{
    var p,q := i,j;
    while p != q
        decreases q-p;
        invariant i <= p <= q <= j;
        invariant forall r | i <= r < p :: a[r] < x;
        invariant forall r | q <= r < j :: a[r] > x;
    {
        var m := p+(q-p)/2;
        if a[m] == x { return m; }
        if a[m] < x { p := m+1; }
        else        { q := m;   }
    }
    k := -p-1;
}
```

Helmingunarleit án lykkju

```
method Search1000( a: array<int>, x: int ) returns ( k: int )
    requires a.Length >= 1000;
    requires forall p,q | 0 <= p < q < 1000 :: a[p] <= a[q];
    ensures 0 <= k <= 1000;
    ensures forall r | 0 <= r < k :: a[r] < x;
    ensures forall r | k <= r < 1000 :: a[r] >= x;
{
    k := 0;
    if a[500] < x       { k := 489;     }
    if a[k+255] < x     { k := k+256;   }
    if a[k+127] < x     { k := k+128;   }
    if a[k+63] < x      { k := k+64;    }
    if a[k+31] < x      { k := k+32;    }
    if a[k+15] < x      { k := k+16;    }
    if a[k+7] < x       { k := k+8;     }
    if a[k+3] < x       { k := k+4;     }
    if a[k+1] < x       { k := k+2;     }
    if a[k] < x         { k := k+1;     }
}
```

$c = 1000$

$c = 511$

$c = 255$

$c = 127$

$c = 63$

$c = 31$

$c = 15$

$c = 7$

$c = 3$

$c = 1$

$c = 0$

Stöðulýsing:

$< x$	?	$\geq x$

$0 \quad k \quad k + c \quad 1000$

Fyrir $c =$ 1000,511,255,127,63,31,15,7,3,1,0

Sjá einnig Search1000.dfy í Canvas

Binary Search

- A very fast search method
- More complicated than linear search
- Requires that the values are already sorted
- A very important and common method
- Everyone must know binary search

Binary search, example

- Search for 19 in the ordered sequence 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22

Binary search, example

- Search for 19 in the ordered sequence 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Originally the unknown section is the entire sequence and the sections < 19 and ≥ 19 are empty

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Binary search, example

- Search for 19 in the ordered sequence $1\ 2\ 3\ 5\ 6\ 7\ 8\ 10\ 12\ 13\ 15\ 16\ 18\ 19\ 20\ 22$
- Originally the unknown section is the entire sequence and the sections < 19 and ≥ 19 are empty

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The sequence contains 16 values and the middle position is therefore position 8 which contains $10 < 19$

Binary search, example

- Search for 19 in the ordered sequence 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Originally the unknown section is the entire sequence and the sections < 19 and ≥ 19 are empty

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The sequence contains 16 values and the middle position is therefore position 8 which contains $10 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Binary search, example

- Search for 19 in the ordered sequence 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Originally the unknown section is the entire sequence and the sections < 19 and ≥ 19 are empty

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The sequence contains 16 values and the middle position is therefore position 8 which contains $10 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The next middle position is position 12 which contains $16 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Binary search, example

- Search for 19 in the ordered sequence 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Originally the unknown section is the entire sequence and the sections < 19 and ≥ 19 are empty

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The sequence contains 16 values and the middle position is therefore position 8 which contains $10 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The next middle position is position 12 which contains $16 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The next middle position is position 14 which contains $19 \geq 19$

Binary search, example

- Search for 19 in the ordered sequence 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Originally the unknown section is the entire sequence and the sections < 19 and ≥ 19 are empty

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The sequence contains 16 values and the middle position is therefore position 8 which contains $10 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The next middle position is position 12 which contains $16 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The next middle position is position 14 which contains $19 \geq 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

Binary search, example

- Search for 19 in the ordered sequence 1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22
- Originally the unknown section is the entire sequence and the sections < 19 and ≥ 19 are empty

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The sequence contains 16 values and the middle position is therefore position 8 which contains $10 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The next middle position is position 12 which contains $16 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The next middle position is position 14 which contains $19 \geq 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

- The next middle position is position 18 which contains $18 < 19$

1	2	3	5	6	7	8	10	12	13	15	16	18	19	20	22

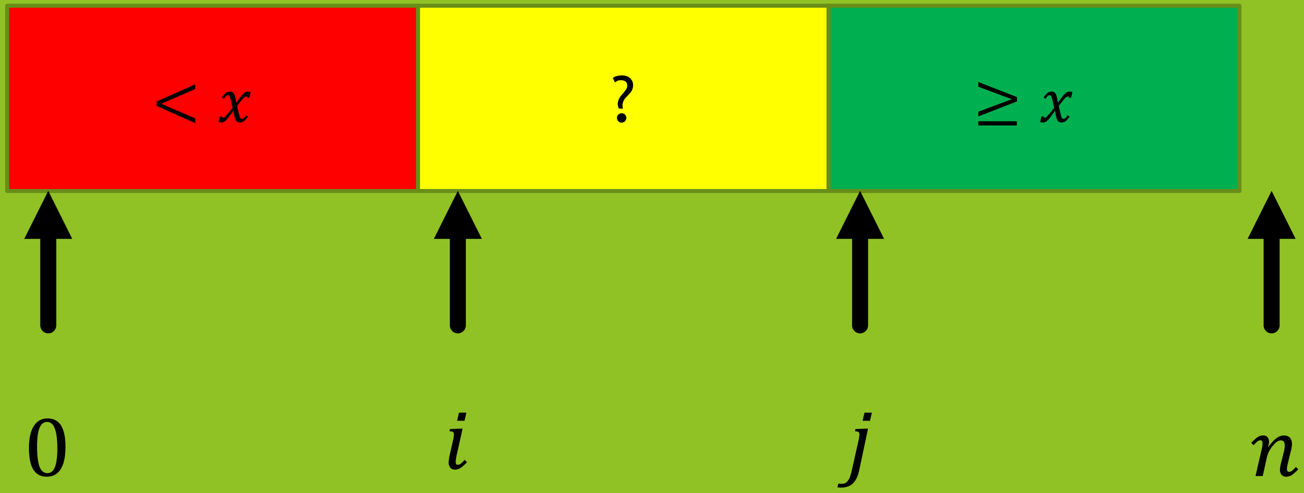
Fundamental idea of binary search

Loop invariant:

$< x$	$?$	$\geq x$

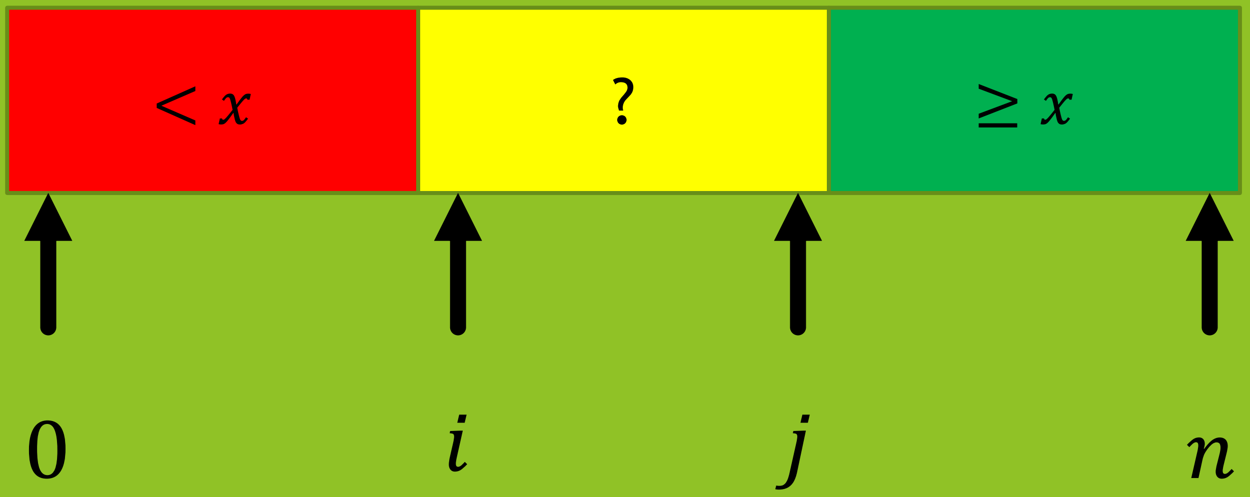
Detailed implementation

Loop invariant:



Various other possibilities

Loop invariant:



Binary search

{
Usage: $i := \text{search}(x, a_0, a_1, \dots, a_{n-1})$
Pre: x is an integer,
$a_0, a_1, \dots, a_{n-1}$ are integers in ascending order
Post: $0 \le i \le n,\ \ a_0, \dots, a_{i-1} < x \le a_i, \dots, a_{n-1}$
}

function search(x : int, $a_0, a_1, \dots, a_{n-1}$: int)
???

Binary search

{

Usage: i := search(x, $a_0, a_1, \ldots, a_{n-1}$)

Pre: x is an integer,

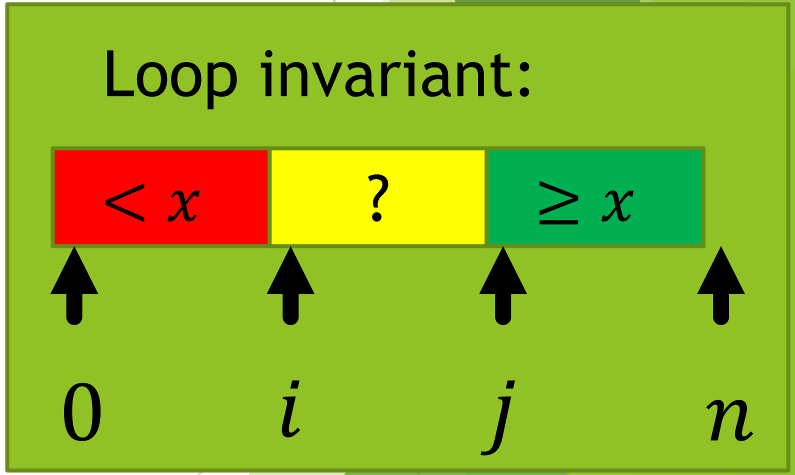
$a_0, a_1, \ldots, a_{n-1}$ are integers in ascending order

Post: $0 \le i \le n$, $a_0, \ldots, a_{i-1} < x \le a_i, \ldots, a_{n-1}$

}

function search(x : int, $a_1, a_2, \ldots, a_{n-1}$: int)

???

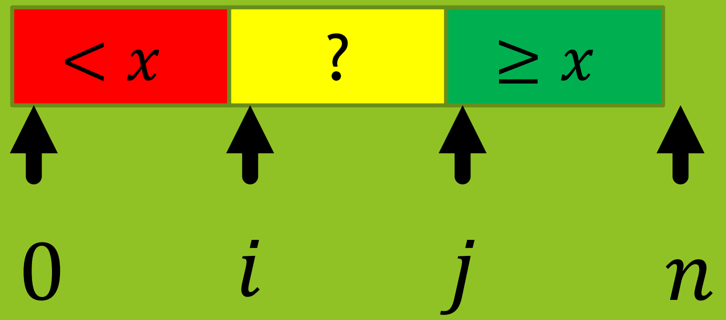


Binary search

{
Usage: i := search($x, a_0, a_1, \ldots, a_{n-1}$)
Pre: x is an integer,
$a_0, a_1, \ldots, a_{n-1}$ are integers in ascending order
Post: $0 \le i \le n$, $a_0, \ldots, a_{i-1} < x \le a_i, \ldots, a_{n-1}$
}

function search(x : int, $a_1, a_2, \ldots, a_{n-1}$: int)
    ???
    while ???
        { $0 \le i \le j \le n$, $a_0, \ldots, a_{i-1} < x \le a_j, \ldots, a_{n-1}$ }
        ???
    ???

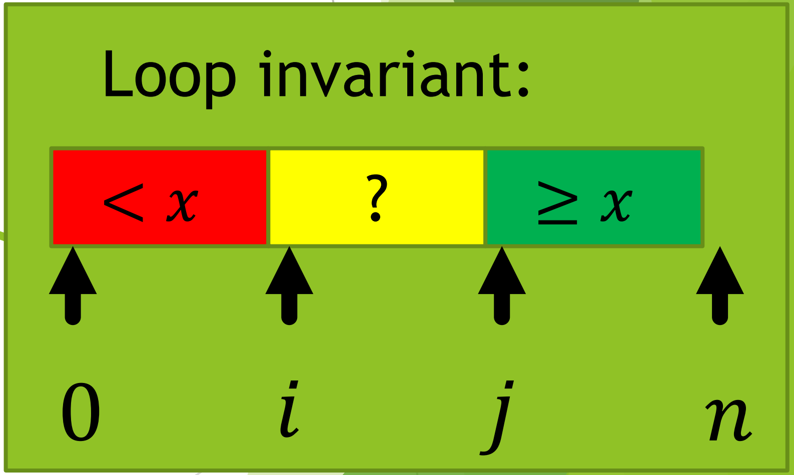
Loop invariant:



Binary search

{
Usage: $i := \text{search}(x, a_0, a_1, \ldots, a_{n-1})$
Pre: x is an integer,
$a_0, a_1, \ldots, a_{n-1}$ are integers in ascending order
Post: $0 \le i \le n$, $a_0, \ldots, a_{i-1} < x \le a_i, \ldots, a_{n-1}$
}

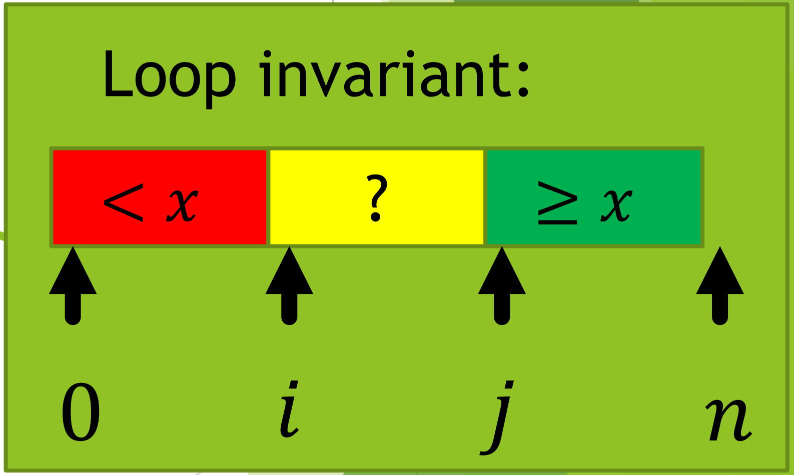
function search(x : int, $a_1, a_2, \ldots, a_{n-1}$: int)
    ???
    while $i \neq j$
        { $0 \le i \le j \le n$, $a_0, \ldots, a_{i-1} < x \le a_j, \ldots, a_{n-1}$ }
        ???
    ???



Binary search

{
Usage: i := search($x, a_0, a_1, \dots, a_{n-1}$)
Pre: x is an integer,
$a_0, a_1, \dots, a_{n-1}$ are integers in ascending order
Post: $0 \le i \le n$, $a_0, \dots, a_{i-1} < x \le a_i, \dots, a_{n-1}$
}

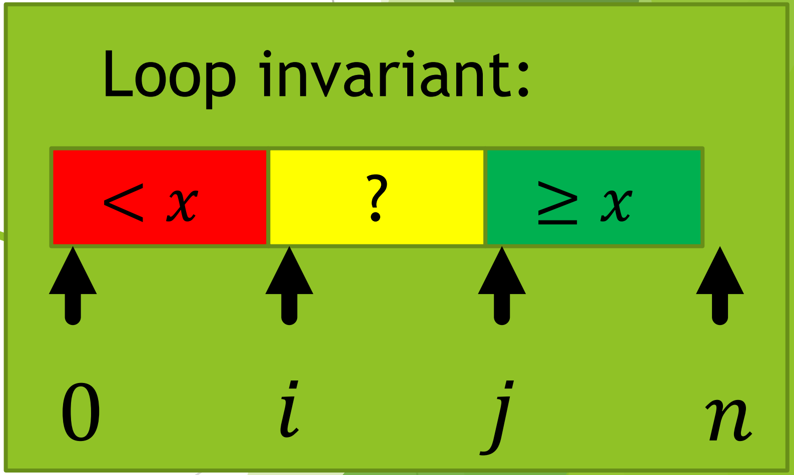
```
function search( x : int, a_1, a_2, ..., a_{n-1}: int )
    ???
    while i ≠ j
        { 0 ≤ i ≤ j ≤ n, a_0, ..., a_{i-1} < x ≤ a_j, ..., a_{n-1} }
        ???
    return i
```



Binary search

{
Usage: i := search(x, $a_0, a_1, \ldots, a_{n-1}$)
Pre: x is an integer,
$a_0, a_1, \ldots, a_{n-1}$ are integers in ascending order
Post: $0 \leq i \leq n$, $a_0, \ldots, a_{i-1} < x \leq a_i, \ldots, a_{n-1}$
}

function search(x : int, $a_1, a_2, \ldots, a_{n-1}$: int)
    i := 0; j := n
    while $i \neq j$
        { $0 \leq i \leq j \leq n$, $a_0, \ldots, a_{i-1} < x \leq a_j, \ldots, a_{n-1}$ }
        ???
    return i



Binary search

{
Usage: i := search(x, $a_0, a_1, \ldots, a_{n-1}$)
Pre: x is an integer,
$a_0, a_1, \ldots, a_{n-1}$ are integers in ascending order
Post: $0 \le i \le n$, $a_0, \ldots, a_{i-1} < x \le a_i, \ldots, a_{n-1}$
}

function search(x : int, $a_1, a_2, \ldots, a_{n-1}$: int)
  i := 0; j := n
  while $i \ne j$
    { $0 \le i \le j \le n$, $a_0, \ldots, a_{i-1} < x \le a_j, \ldots, a_{n-1}$ }
    m := $\lfloor (i+j)/2 \rfloor$
    if $a_m < x$ **then** i := $m + 1$
    else j := m
  return i

Loop invariant:

$< x$	?	$\ge x$

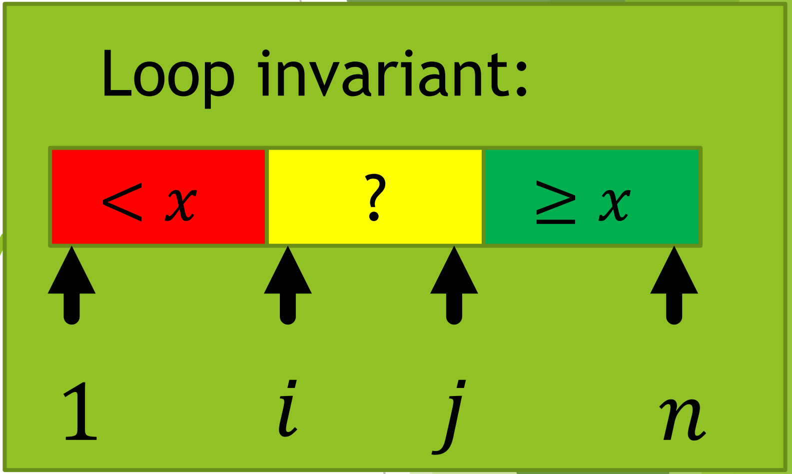
0 i j n

Same as $i + \lfloor (j-i)/2 \rfloor$

Binary search (another version)

{
Usage: i := search(x, $a_1, a_2, \dots, a_n$)
Pre: x is an integer,
$a_1, a_2, \dots, a_n$ are integers in ascending order
Post: $1 \le i \le n+1$, $a_1, \dots, a_{i-1} < x \le a_i, \dots, a_n$
}

function search(x : int, $a_1, a_2, \dots, a_n$: int)
  $i := 1$; $j := n$
  while $i \le j$
    { $1 \le i \le j+1 \le n+1$, $a_1, \dots, a_{i-1} < x \le a_{j+1}, \dots, a_n$ }
    $m := \lfloor (i+j)/2 \rfloor$
    if $a_m < x$ **then** $i := m + 1$
    else $j := m - 1$
  return i



Binary search (still another version)

{

Usage: i := search(x, $a_0, a_1, \ldots, a_{n-1}$)

Pre: x is an integer,

$a_0, a_1, \ldots, a_{n-1}$ are integers in ascending order

Post: $0 \le i < n$ and $a_i = x$ OR

$i < 0$ and $a_0, a_1, \ldots, a_{n-1} \neq x$

}

function search(x : int, $a_1, a_2, \ldots, a_{n-1}$: int)

  i := 0; j := n

  while $i \neq j$

    { $0 \le i \le j \le n$, $a_0, \ldots, a_{i-1} < x < a_j, \ldots, a_{n-1}$ }

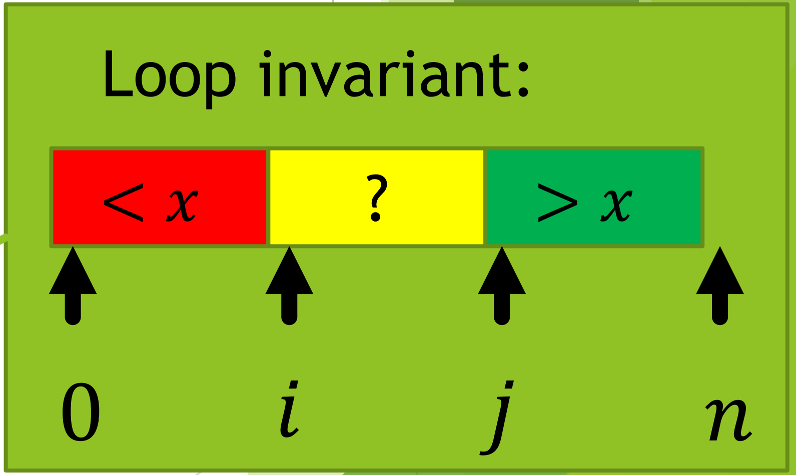
    m := $\lfloor (i + j)/2 \rfloor$

    if $a_m = x$ **then** **return** i

    if $a_m < x$ **then** i := $m + 1$

    else j := m

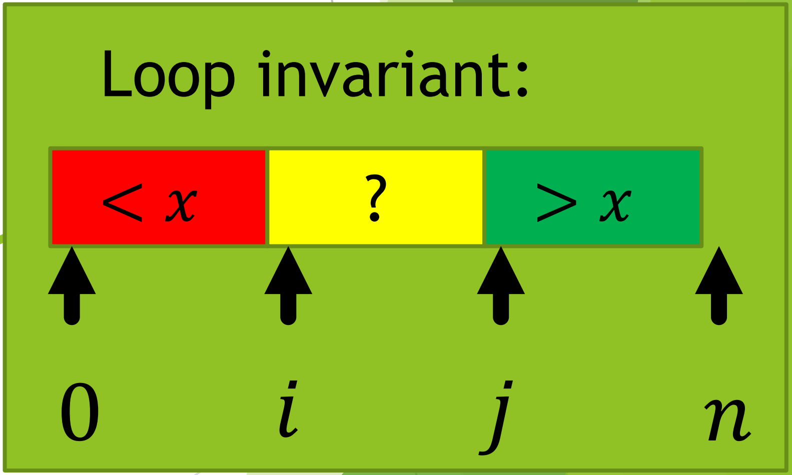
  return -1



Binary search (still another version)

{
Usage: i := search(x, $a_0, a_1, \dots, a_{n-1}$)
pre: x is an integer,
$a_0, a_1, \dots, a_{n-1}$ are integers in ascending order
Post: $0 \le i < n$ and $a_i = x$ OR
$i < 0$ and $a_0, a_1, \dots, a_{n-1} \neq x$
}

function leita(x : int, $a_1, a_2, \dots, a_{n-1}$: int)
    i := 0; j := n
    while $i \neq j$
        { $0 \le i \le j \le n$, $a_0, \dots, a_{i-1} < x < a_j, \dots, a_{n-1}$ }
        m := $\lfloor (i+j)/2 \rfloor$
        if $a_m = x$ **then**   **return** i
        if $a_m < x$ **then** i := $m + 1$
        else j := m
    return $-\mathrm{i} - 1$



Why is this a smart return value?

Binary search (still another version)

{

Usage: i := search(x, $a_0, a_1, \dots, a_{n-1}$)

Pre: x is an integer,

$a_0, a_1, \dots, a_{n-1}$ are integers in ascending order

Post: $0 \le i < n$ and $a_i = x$ OR

$0 \le -i - 1 \le n$ and $a_0, a_1, \dots, a_{-i-2} < x < a_{-i-1}, \dots, a_{n-1}$

}

function leita(x : int, $a_1, a_2, \dots, a_{n-1}$: int)

i := 0; j := n

while $i \ne j$

{ $0 \le i \le j \le n$, $a_0, \dots, a_{i-1} < x < a_j, \dots, a_{n-1}$ }

m := $\lfloor (i+j)/2 \rfloor$

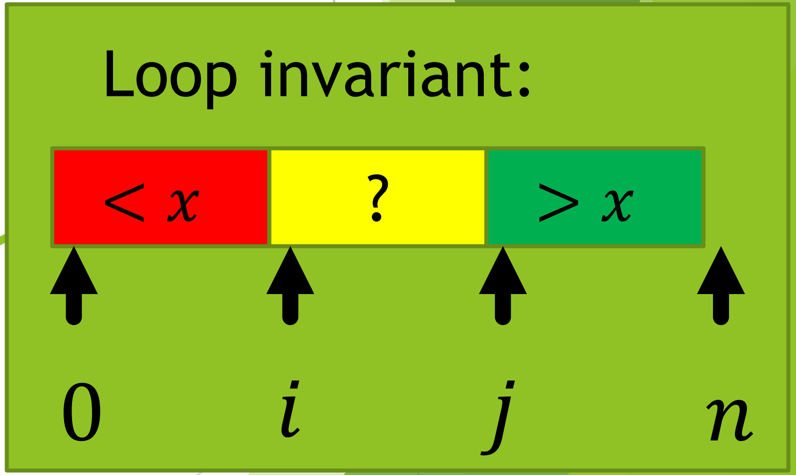
if $a_m = x$ **then** **return** i

if $a_m < x$ **then** i := $m + 1$

else j := m

return $-i - 1$

Why is this a smart return value?



Binary search in Dafny

```
method Search( a: array<int>, i: int, j: int, x: int ) returns( k: int )
    requires 0 <= i <= j <= a.Length;
    requires forall p,q | i <= p < q < j :: a[p] <= a[q];
    ensures i <= k <= j;
    ensures forall r | i <= r < k :: a[r] < x;
    ensures forall r | k <= r < j :: a[r] >= x;
{
    var p,q := i,j;
    while p != q
        invariant i <= p <= q <= j;
        decreases q-p;
        invariant forall r | i <= r < p :: a[r] < x;
        invariant forall r | q <= r < j :: a[r] >= x;
    {
        var m := p+(q-p)/2;
        if a[m] < x { p := m+1; }
        else        { q := m;   }
    }
    k := p;
}
```

See also Search.dfy in Canvas

Binary search in Dafny

```
method Search( a: array<int>, i: int, j: int, x: int ) returns( k: int )
    decreases j-i;
    requires 0 <= i <= j <= a.Length;
    requires forall p,q | i <= p < q < j :: a[p] <= a[q];
    ensures k < 0 ==> !(x in a[i..j]);
    ensures k < 0 ==> i <= -k-1 <= j;
    ensures k < 0 ==> forall r | i <= r < -k-1 :: a[r] < x;
    ensures k < 0 ==> forall r | -k-1 <= r < j :: a[r] > x;
    ensures k >= 0 ==> i <= k < j;
    ensures k >= 0 ==> a[k] == x;
{
    if i == j { return -1-i; }
    var m := i+(j-i)/2;
    if a[m] == x { return m; }
    if a[m] < x { k := Search(a,m+1,j,x); }
    else        { k := Search(a,i,m,x);   }
}
```

Binary search in Dafny

```
method Search( a: array<int>, i: int, j: int, x: int ) returns( k: int )
    requires 0 <= i <= j <= a.Length;
    requires forall p,q | i <= p < q < j :: a[p] <= a[q];
    ensures k < 0 ==> !(x in a[i..j]);
    ensures k < 0 ==> i <= -k-1 <= j;
    ensures k < 0 ==> forall r | i <= r < -k-1 :: a[r] < x;
    ensures k < 0 ==> forall r | -k-1 <= r < j :: a[r] > x;
    ensures k >= 0 ==> i <= k < j;
    ensures k >= 0 ==> a[k] == x;
{
    var p,q := i,j;
    while p != q
        decreases q-p;
        invariant i <= p <= q <= j;
        invariant forall r | i <= r < p :: a[r] < x;
        invariant forall r | q <= r < j :: a[r] > x;
    {
        var m := p+(q-p)/2;
        if a[m] == x { return m; }
        if a[m] < x      { p := m+1; }
        else             { q := m;   }
    }
    k := -p-1;
}
```

Binary search without a loop

```
method Search1000( a: array<int>, x: int ) returns ( k: int )
    requires a.Length >= 1000;
    requires forall p,q | 0 <= p < q < 1000 :: a[p] <= a[q];
    ensures 0 <= k <= 1000;
    ensures forall r | 0 <= r < k :: a[r] < x;
    ensures forall r | k <= r < 1000 :: a[r] >= x;
{
    k := 0;
    if a[500] < x        { k := 489;     }
    if a[k+255] < x      { k := k+256;   }
    if a[k+127] < x      { k := k+128;   }
    if a[k+63] < x       { k := k+64;    }
    if a[k+31] < x       { k := k+32;    }
    if a[k+15] < x       { k := k+16;    }
    if a[k+7] < x        { k := k+8;     }
    if a[k+3] < x        { k := k+4;     }
    if a[k+1] < x        { k := k+2;     }
    if a[k] < x          { k := k+1;     }
}
```

$c = 1000$

$c = 511$

$c = 255$

$c = 127$

$c = 63$

$c = 31$

$c = 15$

$c = 7$

$c = 3$

$c = 1$

$c = 0$

State description:

$< x$	?	$\geq x$

0 $\quad k \quad$ $k + c$ $\quad 1000$

For $c =$
1000,511,255,127,63,31,15,7,3,1,0

See also Search1000.dfy in Canvas