

algorithmic thinking

we need names for the data

- a is the array of ordered data items, indexed starting at 1:

a

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ala	arg	asn	asp	cys	glu	his	ile	leu	lys	met	phe	pro	ser	thr

- $a[1]$ is the first data item, $a[2]$ is the second data item, and so on, up to $a[15]$

- $query$ is the item we are searching for

- **our task:** given $query$, output the index of $query$ in array a

more names

a

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ala	arg	asn	asp	cys	glu	his	ile	leu	lys	met	phe	pro	ser	thr

- $first$ and $last$ refer to the indices bounding the part of the array we are searching,
- $middle$ is the index halfway between $first$ and $last$
- initially, $first$ is 1 and $last$ is 15

$first$	$middle$	$last$
<input style="width: 30px; height: 20px;" type="text" value="1"/>	<input style="width: 30px; height: 20px;" type="text"/>	<input style="width: 30px; height: 20px;" type="text" value="15"/>

binary search algorithm

input: output: index of query in a

$first$	$middle$	$last$
<input style="width: 30px; height: 20px;" type="text" value="1"/>	<input style="width: 30px; height: 20px;" type="text"/>	<input style="width: 30px; height: 20px;" type="text" value="15"/>

search

set $middle$ to be halfway between $first$ and $last$
if $query == a[middle]$ **then** how to do the update?
 output $middle$
else
 update $first$ and $last$
search in a between $first$ and $last$

examples of updating $first$, $last$

a

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ala	arg	asn	asp	cys	glu	his	ile	leu	lys	met	phe	pro	ser	thr

$query$	$first$	$middle$	$last$
<input style="width: 50px; height: 20px;" type="text" value="leu"/>	<input style="width: 30px; height: 20px;" type="text" value="1"/>	<input style="width: 30px; height: 20px;" type="text" value="8"/>	<input style="width: 30px; height: 20px;" type="text" value="15"/>
	$first$	$middle$	$last$
	<input style="width: 30px; height: 20px;" type="text" value="9"/>	<input style="width: 30px; height: 20px;" type="text" value="12"/>	<input style="width: 30px; height: 20px;" type="text" value="15"/>
	$first$	$middle$	$last$
	<input style="width: 30px; height: 20px;" type="text" value="9"/>	<input style="width: 30px; height: 20px;" type="text" value="10"/>	<input style="width: 30px; height: 20px;" type="text" value="11"/>
	$first$	$middle$	$last$
	<input style="width: 30px; height: 20px;" type="text" value="9"/>	<input style="width: 30px; height: 20px;" type="text" value="9"/>	<input style="width: 30px; height: 20px;" type="text" value="9"/>

binary search algorithm

search

$middle = first + (last - first)/2$
if $query == a[middle]$ **then**
 output $middle$
else
if $query < a[middle]$ **then** $last = middle - 1;$
if $query > a[middle]$ **then** $first = middle + 1;$
search in a between $first$ and $last$

binary search algorithm

a 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
ala arg asn asp cys glu his ile leu lys met phe pro ser thr

input: *query*

output: index of *query* in array *a*

first = 1; *last* = 15;

search

middle = *first* + (*last* - *first*)/2

if *query* == *a*[*middle*] **then**

 output *middle*

else

if *query* < *a*[*middle*] **then** *last* = *middle* - 1;

if *query* > *a*[*middle*] **then** *first* = *middle* + 1;

 search in *a* between *first* and *last*

things in a program

- **variables:** data items that may change over time
- **identifiers:** names of variables
- **instructions/statements:** actions on data items
 - compare data values
 - assignment statement: assign a new value to a variable
- **control flow instructions**
 - if ... then ... else
 - while / repeat

input: *query*

output: index of *query* in array *a*

first = 1; *last* = 15;

middle = *first* + (*last* - *first*)/2

if *query* == *a*[*middle*] **then**

 output *middle*

else

if *query* < *a*[*middle*] **then** *last* = *middle* - 1;

if *query* > *a*[*middle*] **then** *first* = *middle* + 1;

 search in *a* between *first* and *last*

real java code for binary search!

```
private int search(int query, int first, int last)
{
    int middle, result;
    middle := (first + last)/2;
    if (query == a[middle]) result = middle;
    else if (query < a[middle])
        result = search(query, first, middle-1);
    else result = search(query, middle+1, last);
    return result;
}
```

food for thought

- Under which conditions (regarding the input data) does our search algorithm work?

(Hint: Think about the number and sequence of entries in the array.)

- How can the algorithm be extended to work in cases where these assumptions don't hold?