

## algorithmic thinking

## Key Learning Goals for Today

After the “algorithmic thinking” unit, you will:

- understand the searching and sorting problems
- be able to describe and simulate the binary search algorithm
- be able to describe at least one sorting algorithm
- appreciate the care that must be taken to precisely express an algorithm
- know some ways to measure algorithms
- believe that of two algorithms for the same problem, each may be better on certain measures

Bonus: a preview of programs and programming terminology

## a search problem

ala arg asn asp cys glu his ile leu lys met phe pro ser thr

Here's a list of “second-level domain names under .com”.  
What does that mean?

How would you find “leu” in the list?

What if the list included all second-level domain names under .com?

## 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 

query
-------
- **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**

1
---

**middle**

--

**last**

15
----

## binary search algorithm

input: 

query
-------

 output: index of query in a

initially: 

first	1
-------	---

middle	
--------	--

last	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'															
	leu														
						<i>first</i>			<i>middle</i>						<i>last</i>
						1			8						15
										<i>first</i>					
						9				12					15
											<i>middle</i>				
										9	10				
												<i>middle</i>			
											9				
													<i>last</i>		
													9		

## 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)/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

identifier/variable

input: query

output: index of query in array a

first = 1; last = 15;

middle = (first + last)/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

control flow statements

assignment statements

## binary search algorithm

**search**

middle = (first + last)/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

## real java code for binary search!

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



## 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?