COMP110: Principles of Computing

5: Computational Complexity

Learning outcomes

- **Explain** the notion of computability
- **Use** “big *O*” notation to express computational complexity
- **Apply** appropriate algorithms to achieve efficiency

Search

Anderson, Martha
Parker, Debra
Russell, Mildred
Stewart, Howard
White, Amanda
Perez, Diana
Lewis, Rose
Scott, Michelle
Davis, Marilyn
Cox, Shirley
Young, Frank
Collins, Jane
Kelly, Philip
Miller, Jeremy
Clark, Stephanie
Brown, Janet
Diaz, Harold
Hughes, Aaron
Sanders, Phillip
Williams, Billy
Henderson, Lawrence
Baker, Theresa
Gonzalez, Adam
Lopez, Jeffrey
Ward, Jessica

- We have a list of names, each with some data associated

Search

Anderson, Martha
Parker, Debra
Russell, Mildred
Stewart, Howard
White, Amanda
Perez, Diana
Lewis, Rose
Scott, Michelle
Davis, Marilyn
Cox, Shirley
Young, Frank
Collins, Jane
Kelly, Philip
Miller, Jeremy
Clark, Stephanie
Brown, Janet
Diaz, Harold
Hughes, Aaron
Sanders, Phillip
Williams, Billy
Henderson, Lawrence
Baker, Theresa
Gonzalez, Adam
Lopez, Jeffrey
Ward, Jessica

- We have a list of names, each with some data associated
- We want to find one of them

Linear search

Anderson, Martha
Parker, Debra
Russell, Mildred
Stewart, Howard
White, Amanda
Perez, Diana
Lewis, Rose
Scott, Michelle
Davis, Marilyn
Cox, Shirley
Young, Frank
Collins, Jane
Kelly, Philip
Miller, Jeremy
Clark, Stephanie
Brown, Janet
Diaz, Harold
Hughes, Aaron
Sanders, Phillip
Williams, Billy
Henderson, Lawrence
Baker, Theresa
Gonzalez, Adam
Lopez, Jeffrey
Ward, Jessica

```
procedure FIND(name, list)
```

Linear search

Anderson, Martha
Parker, Debra
Russell, Mildred
Stewart, Howard
White, Amanda
Perez, Diana
Lewis, Rose
Scott, Michelle
Davis, Marilyn
Cox, Shirley
Young, Frank
Collins, Jane
Kelly, Philip
Miller, Jeremy
Clark, Stephanie
Brown, Janet
Diaz, Harold
Hughes, Aaron
Sanders, Phillip
Williams, Billy
Henderson, Lawrence
Baker, Theresa
Gonzalez, Adam
Lopez, Jeffrey
Ward, Jessica

```
procedure FIND(name, list)
    for each item in list do
```

Linear search

Anderson, Martha
Parker, Debra
Russell, Mildred
Stewart, Howard
White, Amanda
Perez, Diana
Lewis, Rose
Scott, Michelle
Davis, Marilyn
Cox, Shirley
Young, Frank
Collins, Jane
Kelly, Philip
Miller, Jeremy
Clark, Stephanie
Brown, Janet
Diaz, Harold
Hughes, Aaron
Sanders, Phillip
Williams, Billy
Henderson, Lawrence
Baker, Theresa
Gonzalez, Adam
Lopez, Jeffrey
Ward, Jessica

```
procedure FIND(name, list)
    for each item in list do
        if item.name = name then
```

Linear search

Anderson, Martha
Parker, Debra
Russell, Mildred
Stewart, Howard
White, Amanda
Perez, Diana
Lewis, Rose
Scott, Michelle
Davis, Marilyn
Cox, Shirley
Young, Frank
Collins, Jane
Kelly, Philip
Miller, Jeremy
Clark, Stephanie
Brown, Janet
Diaz, Harold
Hughes, Aaron
Sanders, Phillip
Williams, Billy
Henderson, Lawrence
Baker, Theresa
Gonzalez, Adam
Lopez, Jeffrey
Ward, Jessica

```
procedure FIND(name, list)
    for each item in list do
        if item.name = name then
            return item
```

Linear search

Anderson, Martha
Parker, Debra
Russell, Mildred
Stewart, Howard
White, Amanda
Perez, Diana
Lewis, Rose
Scott, Michelle
Davis, Marilyn
Cox, Shirley
Young, Frank
Collins, Jane
Kelly, Philip
Miller, Jeremy
Clark, Stephanie
Brown, Janet
Diaz, Harold
Hughes, Aaron
Sanders, Phillip
Williams, Billy
Henderson, Lawrence
Baker, Theresa
Gonzalez, Adam
Lopez, Jeffrey
Ward, Jessica

```
procedure FIND(name, list)
    for each item in list do
        if item.name = name then
            return item
        end if
    end for
    throw "Not found"
```

Linear search

Anderson, Martha
Parker, Debra
Russell, Mildred
Stewart, Howard
White, Amanda
Perez, Diana
Lewis, Rose
Scott, Michelle
Davis, Marilyn
Cox, Shirley
Young, Frank
Collins, Jane
Kelly, Philip
Miller, Jeremy
Clark, Stephanie
Brown, Janet
Diaz, Harold
Hughes, Aaron
Sanders, Phillip
Williams, Billy
Henderson, Lawrence
Baker, Theresa
Gonzalez, Adam
Lopez, Jeffrey
Ward, Jessica

```
procedure FIND(name, list)
    for each item in list do
        if item.name = name then
            return item
        end if
    end for
    throw "Not found"
end procedure
```

How long does it take?

Socrative room code: `FALCOMPED`

- Suppose there are 25 items in the list

How long does it take?

Socrative room code: `FALCOMPED`

- Suppose there are 25 items in the list
- In the **best case**, how many items do we need to visit before finding the one we want?

How long does it take?

Socrative room code: `FALCOMPED`

- Suppose there are 25 items in the list
- In the **best case**, how many items do we need to visit before finding the one we want?
- How about in the **worst case**?

How long does it take?

Socrative room code: `FALCOMPED`

- If there are 25 items in the list, the **worst case** number of items visited is 25

How long does it take?

Socrative room code: `FALCOMPED`

- If there are 25 items in the list, the **worst case** number of items visited is 25
- How about if there are 50 items?

How long does it take?

Socrative room code: `FALCOMPED`

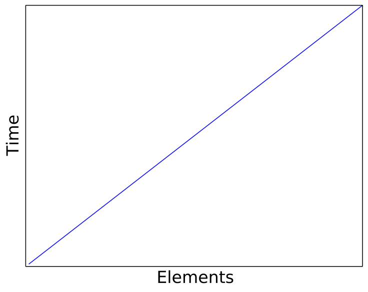
- If there are 25 items in the list, the **worst case** number of items visited is 25
- How about if there are 50 items?
- How about 100 items?

How long does it take?

Socrative room code: `FALCOMPED`

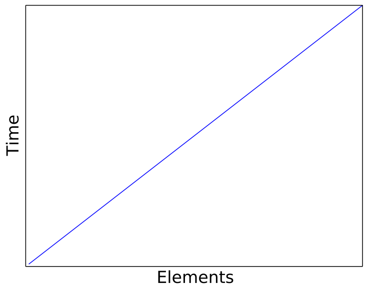
- If there are 25 items in the list, the **worst case** number of items visited is 25
- How about if there are 50 items?
- How about 100 items?
- If the number of items **doubles**, what happens to the amount of time the search takes?

Linear time



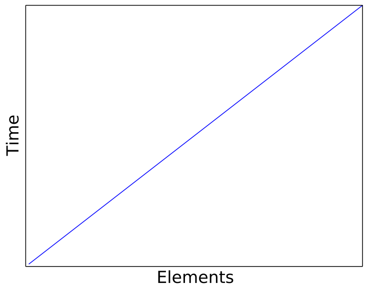
- The running time of linear search is **proportional** to the size *n* of the list

Linear time



- The running time of linear search is **proportional** to the size *n* of the list
- Linear search is said to have **linear time complexity**

Linear time



- The running time of linear search is **proportional** to the size *n* of the list
- Linear search is said to have **linear time complexity**
- Also written as $O(n)$ **time complexity**

Searching a sorted list

Anderson, Martha
Baker, Theresa
Brown, Janet
Clark, Stephanie
Collins, Jane
Cox, Shirley
Davis, Marilyn
Diaz, Harold
Gonzalez, Adam
Henderson, Lawrence
Hughes, Aaron
Kelly, Philip
Lewis, Rose
Lopez, Jeffrey
Miller, Jeremy
Parker, Debra
Perez, Diana
Russell, Mildred
Sanders, Phillip
Scott, Michelle
Stewart, Howard
Ward, Jessica
White, Amanda
Williams, Billy
Young, Frank

- If the list is **sorted** in alphabetical order, we can do better than linear...

Binary search

```
procedure FIND(name, list)
```

Binary search

```
procedure FIND(name, list)
    if list is empty then
        throw "Not found"
    end if
```

Binary search

```
procedure FIND(name, list)
    if list is empty then
        throw "Not found"
    end if
    mid ← the "middle" item of the list
```

Binary search

```
procedure FIND(name, list)
    if list is empty then
        throw "Not found"
    end if
    mid ← the "middle" item of the list
    if name = mid.name then
        return mid
```

Binary search

```
procedure FIND(name, list)
    if list is empty then
        throw "Not found"
    end if
    mid $\leftarrow$ the "middle" item of the list
    if name $=$ mid.name then
        return mid
    else if name $<$ mid.name then
        return FIND(name, first half of list)
```

Binary search

```
procedure FIND(name, list)
    if list is empty then
        throw “Not found”
    end if
    mid $\leftarrow$ the “middle” item of the list
    if name $=$ mid.name then
        return mid
    else if name $<$ mid.name then
        return FIND(name, first half of list)
    else if name $>$ mid.name then
        return FIND(name, second half of list)
```

Binary search

```
procedure FIND(name, list)
    if list is empty then
        throw "Not found"
    end if
    mid $\leftarrow$ the "middle" item of the list
    if name $=$ mid.name then
        return mid
    else if name $<$ mid.name then
        return FIND(name, first half of list)
    else if name $>$ mid.name then
        return FIND(name, second half of list)
    end if
end procedure
```

Find “Lopez, Jeffrey“

Anderson, Martha
Baker, Theresa
Brown, Janet
Clark, Stephanie
Collins, Jane
Cox, Shirley
Davis, Marilyn
Diaz, Harold
Gonzalez, Adam
Henderson, Lawrence
Hughes, Aaron
Kelly, Philip
⟶ Lewis, Rose
Lopez, Jeffrey
Miller, Jeremy
Parker, Debra
Perez, Diana
Russell, Mildred
Sanders, Phillip
Scott, Michelle
Stewart, Howard
Ward, Jessica
White, Amanda
Williams, Billy
Young, Frank

Find “Lopez, Jeffrey”

Anderson, Martha
Baker, Theresa
Brown, Janet
Clark, Stephanie
Collins, Jane
Cox, Shirley
Davis, Marilyn
Diaz, Harold
Gonzalez, Adam
Henderson, Lawrence
Hughes, Aaron
Kelly, Philip
Lewis, Rose
Lopez, Jeffrey
Miller, Jeremy
Parker, Debra
Perez, Diana
Russell, Mildred
⟶ Sanders, Phillip
Scott, Michelle
Stewart, Howard
Ward, Jessica
White, Amanda
Williams, Billy
Young, Frank

Find “Lopez, Jeffrey“

Anderson, Martha
Baker, Theresa
Brown, Janet
Clark, Stephanie
Collins, Jane
Cox, Shirley
Davis, Marilyn
Diaz, Harold
Gonzalez, Adam
Henderson, Lawrence
Hughes, Aaron
Kelly, Philip
Lewis, Rose
Lopez, Jeffrey
Miller, Jeremy
⟶ Parker, Debra
Perez, Diana
Russell, Mildred
Sanders, Phillip
Scott, Michelle
Stewart, Howard
Ward, Jessica
White, Amanda
Williams, Billy
Young, Frank

Find “Lopez, Jeffrey“

Anderson, Martha
Baker, Theresa
Brown, Janet
Clark, Stephanie
Collins, Jane
Cox, Shirley
Davis, Marilyn
Diaz, Harold
Gonzalez, Adam
Henderson, Lawrence
Hughes, Aaron
Kelly, Philip
Lewis, Rose
⟶ Lopez, Jeffrey
Miller, Jeremy
Parker, Debra
Perez, Diana
Russell, Mildred
Sanders, Phillip
Scott, Michelle
Stewart, Howard
Ward, Jessica
White, Amanda
Williams, Billy
Young, Frank

How long does it take?

Socrative room code: `FALCOMPED`

- Each iteration cuts the list in **half**

How long does it take?

Socrative room code: `FALCOMPED`

- Each iteration cuts the list in **half**
- Worst case: we have to keep halving until we get down to a single element

How long does it take?

Socrative room code: `FALCOMPED`

- Each iteration cuts the list in **half**
- Worst case: we have to keep halving until we get down to a single element
- If the size of the list is **doubled**, what happens to the worst-case **number of iterations** required?

How long does it take?

Socrative room code: `FALCOMPED`

- Each iteration cuts the list in **half**
- Worst case: we have to keep halving until we get down to a single element
- If the size of the list is **doubled**, what happens to the worst-case **number of iterations** required?
- **Answer:** it increases by 1

How long does it take?

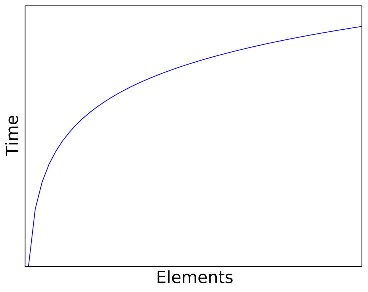
Socrative room code: `FALCOMPED`

- Each iteration cuts the list in **half**
- Worst case: we have to keep halving until we get down to a single element
- If the size of the list is **doubled**, what happens to the worst-case **number of iterations** required?
- **Answer:** it increases by 1
- The running time is **logarithmic** or $O(\log n)$

How long does it take?

Socrative room code: `FALCOMPED`

- Each iteration cuts the list in **half**
- Worst case: we have to keep halving until we get down to a single element
- If the size of the list is **doubled**, what happens to the worst-case **number of iterations** required?
- **Answer:** it increases by 1
- The running time is **logarithmic** or $O(\log n)$



Hidden complexity

```
if name $<$ mid.name then
    return FIND(name, first half of list)
else if name $>$ mid.name then
    return FIND(name, second half of list)
end if
```

Hidden complexity

```
if name < mid.name then
    return FIND(name, first half of list)
else if name > mid.name then
    return FIND(name, second half of list)
end if
```

- Careful how you implement this!

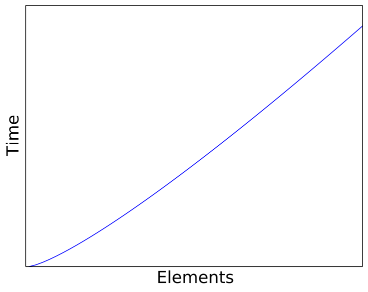
Hidden complexity

```
if name < mid.name then
    return FIND(name, first half of list)
else if name > mid.name then
    return FIND(name, second half of list)
end if
```

- Careful how you implement this!
- **Copying** (half of) a list is **linear** $O(n)$

Hidden complexity

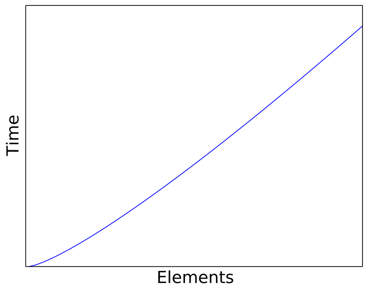
```
if name $<$ mid.name then
    return FIND(name, first half of list)
else if name $>$ mid.name then
    return FIND(name, second half of list)
end if
```



- Careful how you implement this!
- **Copying** (half of) a list is **linear** $O(n)$
- The actual running time would be $O(n \log n)$

Hidden complexity

```
**if** name $<$ mid.name **then**
    **return** FIND(name, first half of list)
**else if** name $>$ mid.name **then**
    **return** FIND(name, second half of list)
**end if**
```



- Careful how you implement this!
- **Copying** (half of) a list is **linear** $O(n)$
- The actual running time would be $O(n \log n)$
- Use **pointers** into the list instead of copying

FALMOUTH UNIVERSITY

Binary search done wrong

```
def binary_search(name, mylist):
    if mylist == []:
        raise ValueError("Not found")

    mid_index = len(mylist) / 2
    mid = mylist[mid_index]

    if name == mid:
        return mid
    elif name < mid:
        return binary_search(name, mylist[:mid_index])
    else:
        return binary_search(name, mylist[mid_index ↩
            +1:])
```

Binary search done right

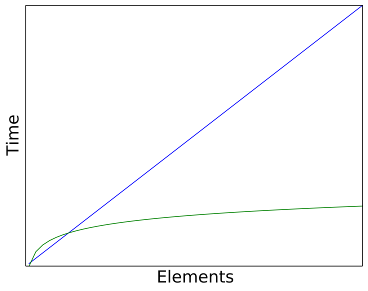
```
def binary_search(name, mylist, start, end):
    if end <= start:
        raise ValueError("Not found")

    mid_index = (start + end) / 2
    mid = mylist[mid_index]

    if name == mid:
        return mid
    elif name < mid:
        return binary_search(name, mylist, start, mid)
    else:
        return binary_search(name, mylist, mid+1, end)
```

Binary search vs linear search

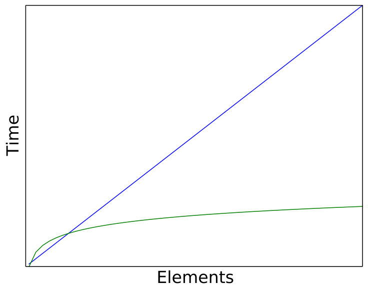
Socrative room code: `FALCOMPED`



- So binary search is better than linear search... right?

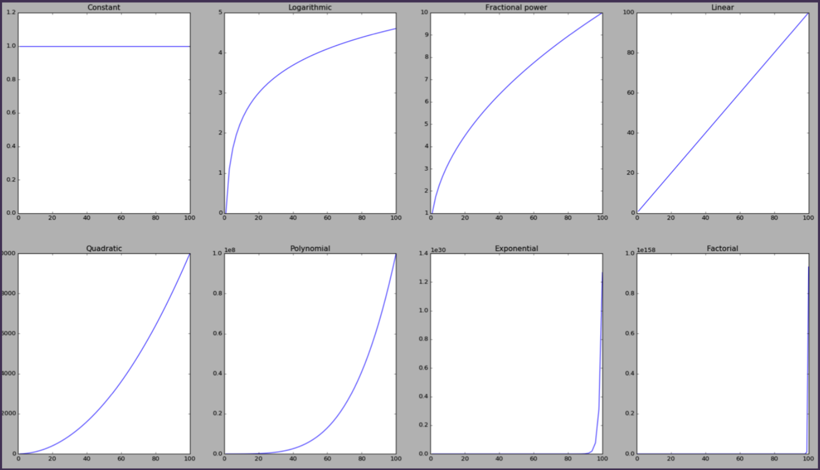
Binary search vs linear search

Socrative room code: `FALCOMPED`



- So binary search is better than linear search... right?
- Discuss in **pairs**
- On Socrative, post **one reason** why, or **one situation** where, linear search may be a better choice than binary search

More complexity classes



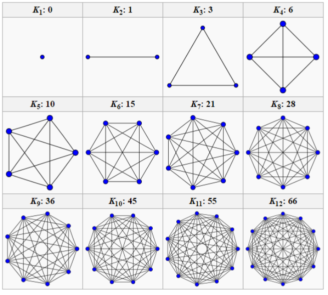
Complexity in games

- Example: collision detection between *n* objects

Complexity in games

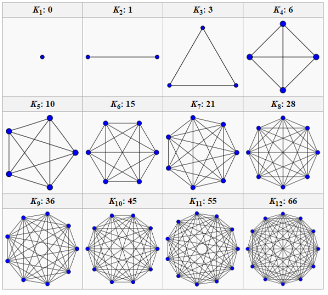
- Example: collision detection between *n* objects
- The naïve way: check **each pair** of objects to see whether they have collided

Complexity in games



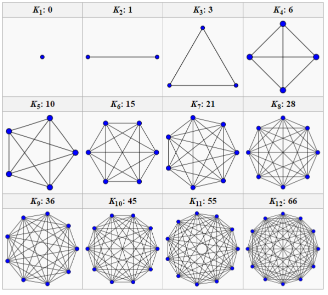
- Example: collision detection between *n* objects
- The naïve way: check **each pair** of objects to see whether they have collided
- This is **quadratic** or $O(n^2)$

Complexity in games



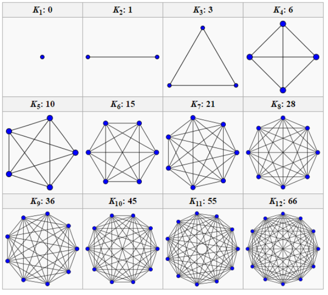
- Example: collision detection between *n* objects
- The naïve way: check **each pair** of objects to see whether they have collided
- This is **quadratic** or $O(n^2)$
- Doubling the number of objects would **quadruple** the time required!

Complexity in games



- Example: collision detection between *n* objects
- The naïve way: check **each pair** of objects to see whether they have collided
- This is **quadratic** or $O(n^2)$
- Doubling the number of objects would **quadruple** the time required!
- Cleverer methods exist that are more scalable

Complexity in games



- Example: collision detection between *n* objects
- The naïve way: check **each pair** of objects to see whether they have collided
- This is **quadratic** or $O(n^2)$
- Doubling the number of objects would **quadruple** the time required!
- Cleverer methods exist that are more scalable
  - Further reading: spatial hashing, quadtrees, octrees, Verlet lists

Caveats

- We've used search as an example, but don't reinvent the wheel!

Caveats

- We’ve used search as an example, but don’t reinvent the wheel!
    - Most languages have existing implementations of linear and binary search on arrays or lists

Caveats

- We've used search as an example, but don't reinvent the wheel!
  - Most languages have existing implementations of linear and binary search on arrays or lists
  - Most languages have a **dictionary** data structure based on **hashing**, which is generally better for this kind of key $\rightarrow$ value look-up

Caveats

- Time complexity only tells us how an algorithm **scales** with the size of the input

Caveats

- Time complexity only tells us how an algorithm **scales** with the size of the input
    - If we know the input will always be **small**, time complexity is not so important

Caveats

- Time complexity only tells us how an algorithm **scales** with the size of the input
  - If we know the input will always be **small**, time complexity is not so important
  - Linear search is quicker than binary search if you only ever have 3 elements

Caveats

- Time complexity only tells us how an algorithm **scales** with the size of the input
  - If we know the input will always be **small**, time complexity is not so important
  - Linear search is quicker than binary search if you only ever have 3 elements
  - Naïve collision detection is fine if your game only ever has 4 objects on screen

Caveats

- Time complexity only tells us how an algorithm **scales** with the size of the input
  - If we know the input will always be **small**, time complexity is not so important
  - Linear search is quicker than binary search if you only ever have 3 elements
  - Naïve collision detection is fine if your game only ever has 4 objects on screen
  - Sometimes complexity in terms of other resources (e.g. space, bandwidth) are more important than time

Caveats

- Time complexity only tells us how an algorithm **scales** with the size of the input
  - If we know the input will always be **small**, time complexity is not so important
  - Linear search is quicker than binary search if you only ever have 3 elements
  - Naïve collision detection is fine if your game only ever has 4 objects on screen
  - Sometimes complexity in terms of other resources (e.g. space, bandwidth) are more important than time
- Software development is all about choosing **the right tool for the job**

Caveats

- Time complexity only tells us how an algorithm **scales** with the size of the input
  - If we know the input will always be **small**, time complexity is not so important
  - Linear search is quicker than binary search if you only ever have 3 elements
  - Naïve collision detection is fine if your game only ever has 4 objects on screen
  - Sometimes complexity in terms of other resources (e.g. space, bandwidth) are more important than time
- Software development is all about choosing **the right tool for the job**
  - If you need scalability, choose a scalable algorithm

Caveats

- Time complexity only tells us how an algorithm **scales** with the size of the input
    - If we know the input will always be **small**, time complexity is not so important
    - Linear search is quicker than binary search if you only ever have 3 elements
    - Naïve collision detection is fine if your game only ever has 4 objects on screen
    - Sometimes complexity in terms of other resources (e.g. space, bandwidth) are more important than time
- Software development is all about choosing **the right tool for the job**
    - If you need scalability, choose a scalable algorithm
    - Otherwise, choose simplicity

Summary

- Time complexity tells us how the running time of an algorithm **scales** with the size of the data it is given

Summary

- Time complexity tells us how the running time of an algorithm **scales** with the size of the data it is given
- Choice of data structures and algorithms can have a large impact on the efficiency of your software

Summary

- Time complexity tells us how the running time of an algorithm **scales** with the size of the data it is given
- Choice of data structures and algorithms can have a large impact on the efficiency of your software
- ... but only if scalability is actually a factor