Let $x$ and $y$ be $N \times 1$ vectors

Let $A$ and $B$ be $N \times N$ and $N \times M$ matrices

How many additions and multiplications to calculate:

- $x^T y$
- $Ax$
- $AB$
- $A^{-1}$
The big-O notation provides an asymptotic upper bound:

\[ O(g(N)) = \{ f : \exists c, N_0 > 0 \text{ s.t. } f(N) \leq cg(N) \forall N > N_0 \} \]
The big-O notation provides an asymptotic upper bound:

\[ O(g(N)) = \{ f : \exists c, N_0 > 0 \text{ s.t. } f(N) \leq c g(N) \forall N > N_0 \} \]

\[ 2N^3 \in O(N^3) \]
\[ N^2 \in O(N^3) \]
\[ N^3 + N^2 \in O(N^3) \]
\[ N^3 + \exp(N) \not\in O(N^3) \]

So is matrix multiplication \( O(N^3) \)? Yes, but: it's also \( O(N^{2.38}) \)!

Conjecture: matrix multiplication is actually \( O(N^2) \).
The big-O notation provides an asymptotic upper bound:

\[ O(g(N)) = \{ f : \exists c, N_0 > 0 \text{ s.t. } f(N) \leq cg(N) \ \forall N > N_0 \} \]

\[ 2N^3 \in O(N^3) \]
\[ N^2 \in O(N^3) \]
\[ N^3 + N^2 \in O(N^3) \]
\[ N^3 + \exp(N) \not\in O(N^3) \]

So is matrix multiplication \( O(N^3) \)? Yes, but: it’s also \( O(N^{2.38}) \)!
The big-O notation provides an asymptotic upper bound:

\[ O(g(N)) = \{ f : \exists c, N_0 > 0 \text{ s.t. } f(N) \leq c g(N) \ \forall N > N_0 \} \]

\[ 2N^3 \in O(N^3) \]
\[ N^2 \in O(N^3) \]
\[ N^3 + N^2 \in O(N^3) \]
\[ N^3 + \exp(N) \not\in O(N^3) \]

So is matrix multiplication \( O(N^3) \)? Yes, but: it’s also \( O(N^{2.38}) \)!

Conjecture: matrix multiplication is actually \( O(N^2) \).
Consider a set of $N$ numbers. We want to sort them in decreasing order. What is the complexity?

Naïve algorithm:

- Find smallest number. Cost? $O(N)$
- Find next smallest number. Cost? $O(N)$
- ...

Overall cost? $O(N^2)$

Can we do better?
A ‘bag’ with three commands: Insert, FindMax and RemoveMax.
A ‘bag’ with three commands: Insert, FindMax and RemoveMax.
A ‘bag’ with three commands: Insert, FindMax and RemoveMax.
A ‘bag’ with three commands: Insert, FindMax and RemoveMax.
A ‘bag’ with three commands: Insert, FindMax and RemoveMax.
A ‘bag’ with three commands: Insert, FindMax and RemoveMax.

Insert: 4 3 12 8 3

FindMax: 6

RemoveMax: 4 = 15
A ‘bag’ with three commands: Insert, FindMax and RemoveMax.
A ‘bag’ with three commands: Insert, FindMax and RemoveMax.
A ‘bag’ with three commands: Insert, FindMax and RemoveMax.
A ‘bag’ with three commands: Insert, FindMax and RemoveMax.

Sorting, clustering, discrete-event simulation, queuing systems
How do we implement this?
PRIORITY QUEUE

Naive approach 1: an unsorted array:

\[(3, 1, 4, 2, 6, 8, 3)\]

What is the cost of \textbf{Insert}?
\[O(1)\]

What is the cost of \textbf{FindMax}?
\[O(N)\]

What is the cost of \textbf{RemoveMax} (assume we've already found the maximum)?
\[O(1)\]

5/15
Naive approach 1: an unsorted array:

\[(3, 1, 4, 2, 6, 8, 3)\]

What is the cost of \textbf{Insert}? \(O(1)\)

What is the cost of \textbf{FindMax}? \(O(N)\)

What is the cost of \textbf{RemoveMax} (assume we’ve already found the maximum)? \(O(1)\)
Naive approach 2: a sorted array:

\[(1, 2, 3, 3, 4, 6, 8)\]
Naive approach 2: a sorted array:

\[(1, 2, 3, 3, 4, 6, 8)\]

Cost of \text{\texttt{FindMax}}? \(O(1)\)
Cost of \text{\texttt{RemoveMax}}? \(O(1)\)
Cost of \text{\texttt{Insert.FindPosition}}? \(O(\log(N)) (\text{binary search})\)
Cost of \text{\texttt{Insert.Insert}}? \(O(N)\)
Naive approach 3: a sorted linked-list:

What is the cost of `FindMax`?

\[ O(1) \]

What is the cost of `RemoveMax`?

\[ O(1) \]

What is the cost of `Insert`?

\[ O(N) \]

Each approach solves one problem, but makes another operation \( \log(N) \). Can we do better?
Naive approach 3: a sorted linked-list:

What is the cost of FindMax? $O(1)$
What is the cost of RemoveMax? $O(1)$
What is the cost of Insert? $O(N)$
Naive approach 3: a sorted linked-list:

What is the cost of \texttt{FindMax}? $O(1)$
What is the cost of \texttt{RemoveMax}? $O(1)$
What is the cost of \texttt{Insert}? $O(N)$

Each approach solves one problem, but makes another operation $\log(N)$. Can we do better?
HEAPS

We need a more complicated data-structure: a Heap.

For a precise definition, see: http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html
For a precise definition, see: http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html
For a precise definition, see: http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html
For a precise definition, see: http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html
Heaps

For a precise definition, see: http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html
For a precise definition, see: http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html
For a precise definition, see: http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html
For a precise definition, see: http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html
For a precise definition, see: http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html
For a precise definition, see: http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html
For a precise definition, see: http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html
HEAPS: Insert
HEAPS: Insert

Insert(9)
HEAPS: Insert

Insert(9)
HEAPS: Insert

Insert(9)
HEAPS: Insert

Insert(9)
HEAPS: Insert

```
2 1 1 3
7
2
4
8
9
```

```
9
8
4
2
2
1
1
3
7
```
HEAPS: Insert

Cost? $O(\log(N))$
HEAPS: RemoveMax

RemoveMax
HEAPS: RemoveMax

RemoveMax

2 1 1 3
7
2
4
8

= 15
HEAPS: RemoveMax

Swap with larger child
HEAPS: RemoveMax
HEAPS: RemoveMax

Cost? $O(\log(N))$
Consider a set of $N$ numbers. Want to sort in decreasing order. Grow a priority queue, sequentially adding elements

- Cost of each step? $O(\log(N))$
- Overall cost? $O(N \log(N))$

Sequentially remove the maximum element

- Cost of each step? $O(\log(N))$
- Overall cost? $O(N \log(N))$

Cost of overall algorithm? $O(N \log(N))$
Quicksort
Quicksort

- Pivot: #7
- Start: #1
- End: #6
• Pivot: #1
• Start: #2
• End: #6
Quicksort

- Pivot: #1
- Start: #2
- End: #6
Quicksort

- Pivot: #2
- Start: #3
- End: #6
Quicksort

- Pivot: #3
- Start: #4
- End: #6
Recurse for each half
Recurse for each half
Quicksort

Recurse for each half
At the end, we have a sorted list

1 2 3 4 7 9 5
Analysis of quicksort

Analysis is a bit harder
What is the worst-case runtime?
What is the best-case runtime?
Average run-time is $\Theta(n \log n)$
Average with respect to what?

Randomized algorithms
Class \( \mathbf{P} \): Problems of polynomial complexity. Let \( T(n) \) be running-time for input size \( n \). Then there is a \( k \) such that:

\[
T(n) = O(n^k)
\]

Class \( \mathbf{E} \): Problems of exponential complexity.

\[
T(n) = \exp(O(n))
\]

Class \( \mathbf{NP} \): Problems of where proposed solution can we verified in polynomial time. E.g. graph isomorphism

Class \( \mathbf{NP} \)-hard: at least as hard as the hardest problems in \( \mathbf{NP} \) (halting problem)

Class \( \mathbf{NP} \)-complete: Hardest problems in \( \mathbf{NP} \) (i.e. problems in both \( \mathbf{NP} \) and \( \mathbf{NP} \)-hard). E.g. travelling salesman.
A million dollar question (literally)