Final exam information

Final exam.
• Check out between 4:30pm on 5/18 and 4:30pm on 5/20.
• 3 hours to complete.
• Gradescope platform.

Rules.
• Honor Code.
• Closed book, closed note.
• Strongly emphasizes material since the midterm.
• Electronic devices are prohibited (except to take the exam).
• 8.5-by-11 cheatsheet (both sides, in your own handwriting).

Final exam preparation.
• Practice by doing old exams.
• Ask questions via Zoom office hours or Ed.
Algorithm Design

- analysis of algorithms
- greedy
- network flow
- dynamic programming
- divide-and-conquer
- randomized algorithms
Algorithm design

Algorithm design patterns.

- Analysis of algorithms.
- Greedy.
- Network flow.
- Dynamic programming.
- Divide-and-conquer.
- Randomized algorithms.

Want more? See COS 340, COS 343, COS 423, COS 445, COS 451, COS 488, ....
Interview questions
ALGORITHM DESIGN

- analysis of algorithms
- greedy
- network flow
- dynamic programming
- divide-and-conquer
- randomized algorithms
Egg drop

**Goal.** Find $T$ using fewest number of tosses.
Egg drop

**Goal.** Find $T$ using fewest number of tosses.

Variant 0. 1 egg.
Variant 1. $\infty$ eggs.
Variant 2. 2 eggs.
Egg drop

**Goal.** Find $T$ using fewest number of tosses.

**Variant 0.** 1 egg.

**Solution.** Use **sequential search**: drop on floors $1, 2, 3, \ldots, T$ until egg breaks.

**Analysis.** 1 egg and $T$ tosses.

running time depends upon a parameter that you don’t no a priori
Egg drop

Goal. Find $T$ using fewest number of tosses.

Variant 1. $\infty$ eggs.

Solution. Binary search for $T$.

- Initialize $[lo, hi] = [0, n]$.
- Repeat until length of interval is 1:
  - drop on floor $mid = (lo + hi) / 2$.
  - if it breaks, update $hi = mid$.
  - if it doesn’t break, update $lo = mid$.

Analysis. $\sim \log_2 n$ eggs, $\sim \log_2 n$ tosses.

Suppose $T$ is much smaller than $n$.
Can you guarantee $\Theta(\log T)$ tosses?
Egg drop

Goal. Find $T$ using fewest number of tosses.

Variant 1’. $\infty$ eggs and $\Theta(\log T)$ tosses.

Solution. Use repeated doubling; then binary search.

- Drop on floors 1, 2, 4, 8, 16, ..., $x$ to find a floor $x$ such that $T \leq x < 2T$.
- Binary search in interval $[\frac{1}{2} x, x]$.

Analysis. $\sim \log_2 T$ eggs, $\sim 2 \log_2 T$ tosses.

- Repeated doubling: 1 egg and $1 + \log_2 x$ tosses.
- Binary search: $\sim \log_2 x$ eggs and $\sim \log_2 x$ tosses.
- Recall: $T \leq x < 2T$. 

<table>
<thead>
<tr>
<th>n</th>
<th>breaks</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>$T$</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Goal. Find $T$ using fewest number of tosses.

Variant 2. 2 eggs.

In worst case, how many tosses needed as a function of $n$?

A. $\Theta(1)$
B. $\Theta(\log n)$
C. $\Theta(\sqrt{n})$
D. $\Theta(n)$
Egg drop (asymmetric search)

Goal. Find $T$ using fewest number of tosses.

Variant 2. 2 eggs.

Solution. Use gridding; then sequential search.
- Toss at floors $\sqrt{n}$, $2\sqrt{n}$, $3\sqrt{n}$, ... until first egg breaks, say at floor $c\sqrt{n}$.
- Sequential search in interval $[c\sqrt{n} - \sqrt{n}, c\sqrt{n}]$.

Analysis. At most $2\sqrt{n}$ tosses.
- First egg: $\leq \sqrt{n}$ tosses.
- Second egg: $\leq \sqrt{n}$ tosses.

Signing bonus 1. Use 2 eggs and at most $\sqrt{2n}$ tosses.

Signing bonus 2. Use 3 eggs and at most $3n^{1/3}$ tosses.
ALGORITHM DESIGN

- analysis of algorithms
- greedy
- network flow
- dynamic programming
- divide-and-conquer
- randomized algorithms

https://algs4.cs.princeton.edu
Greedy algorithms

Make locally optimal choices at each step.

**Familiar examples.**
- Huffman coding.
- Prim’s algorithm.
- Kruskal’s algorithm.
- Dijkstra’s algorithm.

**More classic examples.**
- U.S. coin changing.
- Activity scheduling.
- Gale–Shapley stable marriage.
- ...

**Caveat.** Greedy algorithm rarely leads to globally optimal solution. (but is often used anyway, especially for intractable problems)
Given a document that is a sequence of $n$ words, and a query that is a sequence of $m$ words, find the smallest range in the document that includes the $m$ query words (in the same order).

Ex. Query = “textbook programming computer”
Document search

Given a document that is a sequence of $n$ words, and a query that is a sequence of $m$ words, find the smallest range in the document that includes the $m$ query words (in the same order).

Solution.

- For each query word, maintain a queue of positions where it occurs.

```
textbook: 130 200 250 345

programming: 100 180 184 300 315

computer: 120 150 190 290 320 400
```
Document search

Given a document that is a sequence of $n$ words, and a query that is a sequence of $m$ words, find the smallest range in the document that includes the $m$ query words (in the same order).

Solution.

- For each query word, maintain a queue of positions where it occurs.
- Maintain smallest range containing $m$ query words, with first word at $i$.

range: 130 to ???

textbook: 130 200 250 345
  ↑

programming: 100 180 184 300 315
  ↑

computer: 120 150 190 290 320 400
  ↑
Document search

Given a document that is a sequence of \( n \) words, and a query that is a sequence of \( m \) words, find the smallest range in the document that includes the \( m \) query words (in the same order).

Solution.

- For each query word, maintain a queue of positions where it occurs.
- Maintain smallest range containing \( m \) query words, with first word at \( i \).

range: 130 to ???

textbook: 130 200 250 345

programming: 100 180 184 300 315

computer: 120 150 190 290 320 400
Document search

Given a document that is a sequence of $n$ words, and a query that is a sequence of $m$ words, find the smallest range in the document that includes the $m$ query words (in the same order).

Solution.

- For each query word, maintain a queue of positions where it occurs.
- Maintain smallest range containing $m$ query words, with first word at $i$.

range: 130 to ???
Document search

Given a document that is a sequence of $n$ words, and a query that is a sequence of $m$ words, find the smallest range in the document that includes the $m$ query words (in the same order).

Solution.

- For each query word, maintain a queue of positions where it occurs.
- Maintain smallest range containing $m$ query words, with first word at $i$.

range: 130 to 190

- textbook: 130 200 250 345
- programming: 100 180 184 300 315
- computer: 120 150 190 290 320 400
Document search

Given a document that is a sequence of $n$ words, and a query that is a sequence of $m$ words, find the smallest range in the document that includes the $m$ query words (in the same order).

Solution.

- For each query word, maintain a queue of positions where it occurs.
- Maintain smallest range containing $m$ query words, with first word at $i$.

range: 200 to ???

textbook: 130 200 250 345

programming: 100 180 184 300 315

computer: 120 150 190 290 320 400
Document search

Given a document that is a sequence of \( n \) words, and a query that is a sequence of \( m \) words, find the smallest range in the document that includes the \( m \) query words (in the same order).

Solution.

- For each query word, maintain a queue of positions where it occurs.
- Maintain smallest range containing \( m \) query words, with first word at \( i \).

range: 200 to ???

textbook: 130 200 250 345

programming: 100 180 184 300 315

computer: 120 150 190 290 320 400
Document search

Given a document that is a sequence of \( n \) words, and a query that is a sequence of \( m \) words, find the smallest range in the document that includes the \( m \) query words (in the same order).

Solution.

- For each query word, maintain a queue of positions where it occurs.
- Maintain smallest range containing \( m \) query words, with first word at \( i \).

range: 200 to ???

- **textbook:** 130 200 250 345
- **programming:** 100 180 184 300 315
- **computer:** 120 150 190 290 320 400
Document search

Given a document that is a sequence of $n$ words, and a query that is a sequence of $m$ words, find the smallest range in the document that includes the $m$ query words (in the same order).

Solution.

- For each query word, maintain a queue of positions where it occurs.
- Maintain smallest range containing $m$ query words, with first word at $i$.

range: 200 to ???

textbook: 130 200 250 345

programming: 100 180 184 300 315

computer: 120 150 190 290 320 400
Document search

Given a document that is a sequence of $n$ words, and a query that is a sequence of $m$ words, find the smallest range in the document that includes the $m$ query words (in the same order).

Solution.
- For each query word, maintain a queue of positions where it occurs.
- Maintain smallest range containing $m$ query words, with first word at $i$.

range: 200 to 320

textbook: 130 200 250 345
programming: 100 180 184 300 315
computer: 120 150 190 290 320 400
Document search

Given a document that is a sequence of \( n \) words, and a query that is a sequence of \( m \) words, find the smallest range in the document that includes the \( m \) query words (in the same order).

Solution.

- For each query word, maintain a queue of positions where it occurs.
- Maintain smallest range containing \( m \) query words, with first word at \( i \).

range: 250 to 320

- textbook: 130 200 250 345
- programming: 100 180 184 300 315
- computer: 120 150 190 290 320 400
Document search

Given a document that is a sequence of \( n \) words, and a query that is a sequence of \( m \) words, find the smallest range in the document that includes the \( m \) query words (in the same order).

Solution.

- For each query word, maintain a queue of positions where it occurs.
- Maintain smallest range containing \( m \) query words, with first word at \( i \).

\[
\text{range: 345 to ???}
\]

- textbook: 130 200 250 345
- programming: 100 180 184 300 315
- computer: 120 150 190 290 320 400
Document search

Given a document that is a sequence of \( n \) words, and a query that is a sequence of \( m \) words, find the smallest range in the document that includes the \( m \) query words (in the same order).

Solution.

- For each query word, maintain a queue of positions where it occurs.
- Maintain smallest range containing \( m \) query words, with first word at \( i \).

range: 345 to ???

textbook: 130 200 250 345

programming: 100 180 184 300 315

computer: 120 150 190 290 320 400
Document search

Given a document that is a sequence of $n$ words, and a query that is a sequence of $m$ words, find the smallest range in the document that includes the $m$ query words (in the same order).

Solution.

• For each query word, maintain a queue of positions where it occurs.
• Maintain smallest range containing $m$ query words, with first word at $i$.

range: 345 to ???

textbook: 130 200 250 345

programming: 100 180 184 300 315

computer: 120 150 190 290 320 400
What is running time as a function of the number of words $n$ in the input and the number of words $m$ in the query?

Assume $m \leq n$ and that each word is at most, say, 20 characters.

A. $\Theta(\log n)$
B. $\Theta(n)$
C. $\Theta(n \log n)$
D. $\Theta(n^2)$
Algorithm Design

- analysis of algorithms
- greedy
- network flow
- dynamic programming
- divide-and-conquer
- randomized algorithms

https://algs4.cs.princeton.edu
Network flow

Classic problems on graphs and digraphs.

**Familiar examples.**
- Shortest paths.
- Bipartite matching.
- Maxflow and mincut.
- Minimum spanning tree.

**Other classic examples.**
- Minimum-cost arborescence.
- Non-bipartite matching.
- Assignment problem.
- Minimum-cost flow.
- ...

**Applications.** Many many problems can be modeled using network flow.

“reduction”
Shortest path with orange and black edges

**Goal.** Given a digraph, where each edge has a positive weight and is orange or black, find shortest path from $s$ to $t$ that uses at most $k$ orange edges.

\[ k = 0: \ s \rightarrow 1 \rightarrow t \quad (17) \]
\[ k = 1: \ s \rightarrow 3 \rightarrow t \quad (13) \]
\[ k = 2: \ s \rightarrow 2 \rightarrow 3 \rightarrow t \quad (11) \]
\[ k = 3: \ s \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow t \quad (10) \]
Shortest path with orange and black edges

**Goal.** Given a digraph, where each edge has a positive weight and is orange or black, find shortest path from $s$ to $t$ that uses at most $k$ orange edges.

**Solution.** Create $k+1$ copies of the digraph $G_0, G_1, \ldots, G_k$. For each edge $v \rightarrow w$

- Black: add edge from vertex $v$ in graph $G_i$ to vertex $w$ in $G_i$.
- Orange: add edge from vertex $v$ in graph $G_i$ to vertex $w$ in $G_{i+1}$.

![Diagram showing the shortest path with orange and black edges](image-url)
Shortest path with orange and black edges

**Goal.** Given a digraph, where each edge has a positive weight and is orange or black, find shortest path from $s$ to $t$ that uses at most $k$ orange edges.

**Solution.** Create $k+1$ copies of the digraph $G_0, G_1, \ldots, G_k$. For each edge $v \rightarrow w$
- Black: add edge from vertex $v$ in graph $G_i$ to vertex $w$ in $G_i$.
- Orange: add edge from vertex $v$ in graph $G_i$ to vertex $w$ in $G_{i+1}$.
- Find shortest path from $s$ to every copy of $t$ (and choose best).
Algorithm design: quiz 3

What is worst-case running time of algorithm as a function of k, number of vertices V, and number of edges E? Assume E ≥ V.

A. Θ(E log V)
B. Θ(k E)
C. Θ(k E log V)
D. Θ(k^2 E log V)
ALGORITHM DESIGN

- analysis of algorithms
- greedy
- network flow
- dynamic programming
- divide-and-conquer
- randomized algorithms
Dynamic programming

- Break up problem into a series of overlapping subproblems.
- Build up solutions to larger and larger subproblems.
  (caching solutions to subproblems in a table for later reuse)

Familiar examples.
- Shortest paths in DAGs.
- Seam carving.
- Bellman–Ford.

More classic examples.
- Unix diff.
- Viterbi algorithm for hidden Markov models.
- Smith–Waterman for DNA sequence alignment.
- CKY algorithm for parsing context-free grammars.
  ...

THE THEORY OF DYNAMIC PROGRAMMING
RICHARD BELLMAN
House coloring problem

Goal. Paint a row of $n$ houses red, green, or blue so that

- No two adjacent houses have the same color.
- Minimize total cost, where $\text{cost}(i, \text{color})$ is cost to paint $i$ given color.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>9</td>
<td>22</td>
<td>12</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Cost to paint house $i$ the given color

$(3 + 6 + 4 + 8 + 5 + 8 = 34)$
House coloring problem

**Goal.** Paint a row of $n$ houses red, green, or blue so that

- No two adjacent houses have the same color.
- Minimize total cost, where $\text{cost}(i, \text{color})$ is cost to paint $i$ given color.

**Subproblems.**

- $R[i] = \text{min cost to paint houses } 1, \ldots, i \text{ with } i \text{ red.}$
- $G[i] = \text{min cost to paint houses } 1, \ldots, i \text{ with } i \text{ green.}$
- $B[i] = \text{min cost to paint houses } 1, \ldots, i \text{ with } i \text{ blue.}$
- Optimal cost $= \text{min } \{ R[n], G[n], B[n] \}$.

**Dynamic programming recurrence.**

- $R[i + 1] = \text{cost}(i+1, \text{red}) + \text{min } \{ G[i], B[i] \}$
- $G[i + 1] = \text{cost}(i+1, \text{green}) + \text{min } \{ B[i], R[i] \}$
- $B[i + 1] = \text{cost}(i+1, \text{blue}) + \text{min } \{ R[i], G[i] \}$
What is running time of algorithm as a function of $n$?

A. $\Theta(n)$
B. $\Theta(n \log n)$
C. $\Theta(n \log^2 n)$
D. $\Theta(n^2)$
Divide and conquer

- Break up problem into two or more independent subproblems.
- Solve each subproblem recursively.
- Combine solutions to subproblems to form solution to original problem.

Familiar examples.
- Mergesort.
- Quicksort.

More classic examples.
- Closest pair.
- Convolution and FFT.
- Matrix multiplication.
- Integer multiplication.
  ...

Prototypical usage. Turn brute-force $\Theta(n^2)$ algorithm into $\Theta(n \log n)$ one.
Personalized recommendations

Music site tries to match your song preferences with others.

- Your ranking of songs: $0, 1, \ldots, n-1$.
- My ranking of songs: $a_0, a_1, \ldots, a_{n-1}$.
- Music site consults database to find people with similar tastes.

**Kendall-tau distance.** Number of **inversions** between two rankings.

**Inversion.** Songs $i$ and $j$ are inverted if $i < j$, but $a_i > a_j$.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>you</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>me</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

3 inversions: 2–1, 3–1, 7–6
Counting inversions

**Problem.** Given a permutation of length $n$, count the number of inversions.

<table>
<thead>
<tr>
<th>0</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>4</th>
<th>5</th>
<th>7</th>
<th>6</th>
</tr>
</thead>
</table>

3 inversions: 2–1, 3–1, 7–6

**Brute-force $n^2$ algorithm.** For each $i < j$, check if $a_i > a_j$.

**A bit better.** Run insertion sort; return number of exchanges.

**Goal.** $n \log n$ time (or better).
## Counting inversions: divide-and-conquer

<table>
<thead>
<tr>
<th>input</th>
<th>0</th>
<th>4</th>
<th>3</th>
<th>7</th>
<th>9</th>
<th>1</th>
<th>5</th>
<th>8</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
</table>

### count inversions in left subarray

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>3</th>
<th>7</th>
<th>9</th>
<th>1</th>
<th>5</th>
<th>8</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
</table>

4 - 3 = 1

### count inversions in right subarray

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>3</th>
<th>7</th>
<th>9</th>
<th>1</th>
<th>5</th>
<th>8</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
</table>

5 - 2 = 3
8 - 2 = 6
8 - 6 = 2

3 + 2 + 6 = 11

### count inversions with one element in each subarray

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>3</th>
<th>7</th>
<th>9</th>
<th>1</th>
<th>5</th>
<th>8</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
</table>

3 - 1 = 2
3 - 2 = 1
4 - 1 = 3
4 - 2 = 2
7 - 1 = 6
7 - 2 = 5
7 - 5 = 2
7 - 6 = 1
9 - 1 = 8
9 - 2 = 7
9 - 5 = 4
9 - 6 = 3
9 - 8 = 2

13

### output

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>3</th>
<th>7</th>
<th>9</th>
<th>1</th>
<th>5</th>
<th>8</th>
<th>2</th>
<th>6</th>
</tr>
</thead>
</table>

1 + 3 + 13 = 17

This step seems to require $n^2$ time.
Counting inversions: divide-and-conquer

input

| 0 | 4 | 3 | 7 | 9 | 1 | 5 | 8 | 2 | 6 |

count inversions in left subarray and sort

| 0 | 3 | 4 | 7 | 9 | 1 | 5 | 8 | 2 | 6 |

1

count inversions in right subarray and sort

| 0 | 3 | 4 | 7 | 9 | 1 | 2 | 5 | 6 | 8 |

3

count inversions with one element in each sorted subarray

| 0 | 3 | 4 | 7 | 9 | 1 | 2 | 5 | 6 | 8 |

13

and merge into sorted whole

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

1 + 3 + 13 = 17
What is running time of algorithm as a function of n?

A. $\Theta(n)$

B. $\Theta(n \log n)$

C. $\Theta(n \log^2 n)$

D. $\Theta(n^2)$
Algorithm Design

- analysis of algorithms
- greedy
- network flow
- dynamic programming
- divide-and-conquer
- randomized algorithms
Randomized algorithms

Algorithm whose performance (or output) depends on the results of random coin flips.

Familiar examples.
• Quicksort.
• Quickselect.

More classic examples.
• Rabin–Karp substring search.
• Miller–Rabin primality testing.
• Polynomial identity testing.
• Volume of convex body.
• Universal hashing.
• Global min cut.
…
Nuts and bolts

Problem. A disorganized carpenter has a mixed pile of $n$ nuts and $n$ bolts.
- The goal is to find the corresponding pairs of nuts and bolts.
- Each nut fits exactly one bolt and each bolt fits exactly one nut.
- By fitting a nut and a bolt together, the carpenter can see which one is bigger (but cannot directly compare either two nuts or two bolts).

Brute-force algorithm. Compare each bolt to each nut: $\Theta(n^2)$ compares.
Challenge. Design an algorithm that makes $O(n \log n)$ compares.
Nuts and bolts

**Shuffle.** Shuffle the nuts and bolts.

<table>
<thead>
<tr>
<th>bolts</th>
<th>5</th>
<th>3</th>
<th>6</th>
<th>0</th>
<th>9</th>
<th>1</th>
<th>4</th>
<th>8</th>
<th>2</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>nuts</td>
<td>7’</td>
<td>2’</td>
<td>8’</td>
<td>1’</td>
<td>5’</td>
<td>9’</td>
<td>4’</td>
<td>0’</td>
<td>6’</td>
<td>3’</td>
</tr>
</tbody>
</table>

**Partition.**
- Pick leftmost bolt $i$ and compare against all nuts; divide nuts smaller than $i$ from those that are larger than $i$.
- Let $i'$ be the nut that matches bolt $i$. Compare $i'$ against all bolts; divide bolts smaller than $i'$ from those that are larger than $i'$.

<table>
<thead>
<tr>
<th>bolts</th>
<th>3</th>
<th>0</th>
<th>1</th>
<th>4</th>
<th>2</th>
<th>5</th>
<th>6</th>
<th>9</th>
<th>8</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>nuts</td>
<td>2’</td>
<td>1’</td>
<td>4’</td>
<td>0’</td>
<td>3’</td>
<td>5’</td>
<td>7’</td>
<td>8’</td>
<td>9’</td>
<td>6’</td>
</tr>
</tbody>
</table>

**Divide-and-conquer.** Recursively solve two subproblems.
What is the expected running time of algorithm as a function of n?

A. $\Theta(n)$
B. $\Theta(n \log n)$
C. $\Theta(n \log^2 n)$
D. $\Theta(n^2)$

Algorithm design: quiz 6
Hiring bonus. Algorithm that takes $O(n \log n)$ time in the worst case.
Faculty lead preceptors and graduate student AIs.

Undergraduate graders and lab TAs. Apply to be one next semester!

Ed tech. Several developed here at Princeton!
A farewell video (from P04, Fall 2018)

COS 226 P04 Presents..
“Algorithms and data structures are love. Algorithms and data structures are life.”

— anonymous COS 226 student