Algorithm uses random numbers.
Output and/or run-time may depend on random numbers and we must do expected case analysis.

Advantages:
- Practical - can solve some problems more easily/quickly
- Theoretical - with randomness, can we do more in polynomial time? NP-complete problems?
  - Evidence thin so far.

Examples you’ve seen in CS240, 341:
  - quicksort, quickselect, hashing.

Today: review/intro to expected case analysis
  - quickselect - a case where randomness probably helps (a tiny bit)

Quicksort \( S = \langle s_1, s_2 \ldots, s_n \rangle \)

if \( n = 0, 1 \) return \( S \)
else \( i \leftarrow \text{random number in } \{1, \ldots, n\} \) \( s_i \) is our pivot
  \( L \leftarrow \{ s_j : s_j < s_i \} \) size \( l \)
  \( M \leftarrow \{ s_j : s_j = s_i \} \) size \( m \geq 1 \)
  \( B \leftarrow \{ s_j : s_j > s_i \} \)
return \( \langle \text{Quicksort}(L), M, \text{Quicksort}(B) \rangle \)

worst case \( T(n) = T(n-1) + O(n) \)
solves to \( T(n) = \Theta(n^2) \)

Intuition: we “expect” \( s_i \) to be in the middle so
\( T(n) = 2T(\frac{n}{2}) + O(n) \)
solves to \( T(n) = O(n \log n) \)
Note the difference between
- random algorithm, expected case analysis
- average case analysis (can choose i deterministically, assuming all input ordering equally likely)
A randomized alg. makes no assumption about distribution of inputs.

Randomized Algorithms — more formally
- Model: The algorithm can ask for a random no./bit
  \( x \leftarrow \text{rand} \{1 \ldots n\} \) random number in \( 1 \ldots n \)
  \( x \leftarrow \text{rand} \{0,1\} \) random bit.
  in \( O(1) \) time [more sophisticated: count bit operations]
- how to implement
  - pseudo-random number generator
  - true source of randomness
Run time for fixed input depends on random numbers
- random variable

Sample space — all possible outcomes (runs of the alg.) for fixed input.
random variable maps sample space to integer (runtime)

Expected value of a random variable \( X \)
\[
E(X) = \sum_{x} x \Pr(X = x) \quad \text{for discrete } X
\]

Example: biased coin, \( \Pr(H) = \frac{1}{2}, \Pr(T) = \frac{2}{3} \)
What is expected # coin tosses to get a H?
\[
\sum i \Pr(i \text{ tosses}) = 1 \cdot \frac{1}{2} + 2 \left( \frac{2}{3} \right) \cdot \frac{1}{3} + 3 \left( \frac{2}{3} \right)^2 \cdot \frac{1}{3} + \cdots
\]
Properties:
- \( E(X+Y) = E(X) + E(Y) \) (linearity of expectation)
- \( E(cX) = cE(X) \) (c-constant)
- \( E(XY) = E(X) \cdot E(Y) \) if \( X \) and \( Y \) are independent random variables, i.e. \( \Pr[X=x \text{ and } Y=y] = \Pr[X=x] \cdot \Pr[Y=y] \)
- \( X \leq Y \implies E(X) \leq E(Y) \)
  
  So \( \max \{E(X), E(Y)\} \leq E(\max \{X, Y\}) \)

See CLRS Ch.5 and Appendix C for review.
(Note: Ch.5 uses "indicator random variables" but lectures won't)

For randomized alg,

\[ T(I, R) = \text{run time of alg. on input } I \]
  with sequence \( R \) of random numbers
  - random variable

To get a function of \( n \) (to do asymptotic analysis)
  we want
  - expected (average) over \( R \)
  - max over \( I, |I| = n \)

\[ E(T(I, R)) = \text{expected run time on input } I, \]
\[ = \sum_{R} \Pr[R] \cdot T(I, R) \]  \( \text{check that this is equivalent to our def of } E(\_\_) \)

\[ T(n) = \max_{|I|=n} E(T(I, R)) \leq E(\max_{|I|=n} T(I, R)) \]
Analysis of Quicksort.
usual analysis uses recurrences
assume distinct elements. \(|L| = k, |B| = n-k-1\)
\[ T(n) = \frac{1}{n} \sum_{i=0}^{n-1} \left( T(l) + T(n-l-1) \right) + \Theta(n) \]
recursive calls
\(k\) equally likely to be \(0, 1, \ldots, n-1\)
\[ = \frac{2}{n} \sum_{i=0}^{n-1} i + \Theta(n) \quad \text{and solve} \quad \Theta(n \log n) \]

An alternative analysis
let \(X = \# \) comparisons (random variable)
\[ X = \sum_{\text{pairs } u, v} X(u, v) \quad X'(u, v) = \# \text{ comparisons} \]
\(\text{between } u \text{ and } v\)
we want \(E(X) = E(\sum X(u, v)) = \sum E(X'(u, v)) \)
by linearity of expectation.

Consider where \(u\) and \(v\) are during Quicksort
- initially in same part of partition
- at some time they go in separate parts
- never compared after that.
\[ E(X'(u, v)) = 1 \cdot \Pr\{\text{compare } u, v\} + 0 \cdot \Pr\{\text{don't}\} \]
e.g., \(2, 5, 6, 1, 3\)
pivot on 5. \(L = 2, 1, 3\) \(M = 5\) \(B = 6\)
5 is compared with all. But 2, 6 will never be compared.

Consider the step where \(u, v\) go into separate parts.
Suppose

\[
\begin{array}{cccc}
\text{rank} & 1 & \cdots & r & \cdots & r+k & \cdots & n \\
\text{element} & \cdots & u & \cdots & v & \cdots & \end{array}
\]

in same part

pivot must lie between \( u \) and \( v \)

we compare \( u,v \) iff pivot is \( u \) or \( v \)

\[
\Pr \{ \text{compare } u,v \} = \frac{2}{k+1}
\]

Thus

\[
\sum_{r=1}^{n} \sum_{k=1}^{r} \frac{2}{k+1} \leq 2 \sum_{r=1}^{n} \sum_{k=1}^{r} \frac{1}{k}
\]

\[
\leq \sum_{r=1}^{n} O(\log n) = O(n \log n)
\]

Harmonic series
Selection

Input: - set of \( n \) numbers \( S \)
- number \( k \in \{1, \ldots, n\} \)

return \( k \)th smallest element of \( S \)

\( k=1 \) — min
\( k=2 \) — 2nd min
\( k=n \) — max
\( k=\left\lfloor \frac{n}{2} \right\rfloor \) — median.

\underline{QuickSelect} \( (S, k) \)

if \( n \leq \) constant — sort
else
\( i \leftarrow \text{Rand} [1 \ldots n] \) — \( Si \) is pivot.

partition \( S \) into
\( L = \{ s_j : s_j < s_i \} \) — size \( L \)
\( M = \{ s_j : s_j = s_i \} \) — size \( M \)
\( B = \{ s_j : s_j > s_i \} \) — size \( B \)

if \( k \leq L \) then \( \text{QuickSelect} \ (L, k) \)
else if \( k \leq L + M \) then return \( s_i \)
else return \( \text{QuickSelect} \ (B, k-(L+M)) \).

Worst case: \( O(n^2) \) when pivot is always min or max.
Analysis of Quickselect

(Not covered in class — hopefully this is review)

$T(n)$ — random variable — run time of
Quickselect on set of size $n$.

(note: things depend only on rank of pivot)

Want $E(T(n))$

Recursive call is on set of size $l$ or $n-l$
(assuming distinct elements).

For upper bound assume $k$ lies in larger half.

\[
\begin{align*}
100 & \quad \frac{3}{4}n \\
\frac{1}{2}n & \quad \frac{1}{4}n \\
\end{align*}
\]

Prob. of pivot here is $\frac{1}{2}$

When pivot is here, recursion is $\leq T\left(\frac{3}{4}n\right)$

$E(T(n)) \leq \frac{1}{2} E(T\left(\frac{3}{4}n\right)) + \frac{1}{2} E(T(n-1)) + O(n)$

we are assuming $T(i) \leq T(j)$ if $i \leq j$

recurrence

$f(n) \leq \frac{1}{2} f\left(\frac{3}{4}n\right) + \frac{1}{2} f(n-1) + O(n)$

Prove by ind.: $f(n)$ is $O(n)$. 
History
1960 Hoare QuickSelect
# comparisons \(3n + o(n)\) expected
randomization is great!

1973 Blum, Floyd, Tarjan
non-randomized selection in \(O(n)\) time.
\(5.43n + o(n)\) comparisons.

same wrt \(O(n)\),
but better constant for randomized?

1975 Floyd, Rivest
randomized alg. \(1.5n + o(n)\) expected # comparisons

1989 Munro, Cunto any randomized alg. (only using comparisons)
takes at least \(1.5n\) expected # comparisons

1985 lower bound of \(\frac{2n}{3}\) # comparisons
for non-randomized alg.
So randomization probably helps.

best non-randomized bounds:
upper bound '99 \(2.95n\) # comparisons
lower bound '01 \((2+\varepsilon)n\) \(\varepsilon = 2^{-80}\)

we'll see one lower bound
and idea of \(O(n)\) det. alg.,
\[\text{Note: not about randomized alg.}\]
Thm [Blum et al '75]
Finding median requires $\geq 1.5n$ comparisons in worst case.

**Proof**

Let $L = \{\text{elements} < m\}$ \{need $\frac{n-1}{2}$ elmts in each\}

$H = \{\text{elements} > m\}$

**Claim** # LL and HH comparisons $\geq n-1$ always

**Proof**
Each element in LL must lose a comparison

- - - / HH - win - -
(except m)

**Claim** # LH comparisons $\geq \frac{n-1}{2}$ in worst case

**Proof**
alg. asks for comparisons
adversary gives results of comparisons (maliciously)
placing elements in L, H. "setting" elmts

Alg. compares $x, y$
- if $x \& y$ are set - ok
- if $x$ set, $y$ unset
  - if $x \in L$, put $y$ in H
  - if $x \in H$ - - - L
- if $x, y$ unset
  - put one in L, one in H

**But** when $\frac{n-1}{2}$ elmts in L [or H] stop and
put remaining elmts in other set.
The adversary always sets $\geq \frac{n-1}{2}$ elements
Idea of the worst case \( O(n) \) Selection algorithm.

Divide elements into groups of \( \frac{n}{5} \) and find median of each 5.

Recursively find median of medians, \( P \) and use that as a pivot. We are guaranteed in sorted order

\[
\begin{align*}
\frac{3n}{10} & \quad \frac{7n}{10} \\
\text{Piles in here}
\end{align*}
\]

So we get

\[
T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)
\]

and can prove this is \( O(n) \).

This is sloppy - e.g., \( n \) not a power of 5, \( \frac{n}{5} \) even, etc.