1 Probabilistic Analysis
   - Average Case Analysis of Insertion Sort
Analyzing Average Case Time Complexity

Definition

Let $T(n)$ denote the average case time complexity used by an algorithm to solve a problem on an input size $n$. Then:

$$T(n) = \sum_{I \in D_n} P(I) \cdot t(I)$$

- $D_n$ is the set of all input instances of size $n$
- $I$ denotes instance $I$ taking values over sample space $D_n$
- $P(I)$ denotes the probability with which $I$ occurs
- $t(I)$ denotes time it takes to solve problem on input instance $I$
- $\sum_{I \in D_n} P(I) = 1$ for correct analysis
Light Exercise: Average Case Analysis of Insertion Sort

Need a bit of a refresher on expected values and random variables.
Need a bit of a refresher on expected values and random variables
Need a bit of a refresher on *expected values* and *random variables*
Q: What is the expected number of Heads from one coin toss?

Introduce binary random variable $X_H$ to track this number

$$E[X_H] = 1 \cdot P(X_H = 1) + 0 \cdot P(X_H = 0) = 1 \cdot (1/2) + 0 \cdot (1/2) = 1/2$$
Q: What is the expected number of Heads from one coin toss?

Introduce binary random variable $X_H$ to track this number

$$E[X_H] = 1 \cdot P(X_H = 1) + 0 \cdot P(X_H = 0) = 1 \cdot (1/2) + 0 \cdot (1/2) = 1/2$$

*Expected number of H's from one flip of a fair coin is $1/2$.***
Q: What is the expected number of Heads from one coin toss?

Introduce binary random variable $X_H$ to track this number

$$E[X_H] = 1 \cdot P(X_H = 1) + 0 \cdot P(X_H = 0) = 1 \cdot (1/2) + 0 \cdot (1/2) = 1/2$$

*Expected number of H’s from one flip of a fair coin is 1/2.*

Q: What is the expected number of Heads in $n$ tosses of a coin?
Q: What is the expected number of Heads from one coin toss?

Introduce binary random variable $X_H$ to track this number

$$E[X_H] = 1 \cdot P(X_H = 1) + 0 \cdot P(X_H = 0) = 1 \cdot (1/2) + 0 \cdot (1/2) = 1/2$$

*Expected number of H’s from one flip of a fair coin is 1/2.*

Q: What is the expected number of Heads in $n$ tosses of a coin?

Let $X = \sum_{i=1}^{n} X_{H,i}$ be the total number of H’s in $n$ tosses.
Q: What is the expected number of Heads from one coin toss?

Introduce binary random variable $X_H$ to track this number

$$E[X_H] = 1 \cdot P(X_H = 1) + 0 \cdot P(X_H = 0) = 1 \cdot (1/2) + 0 \cdot (1/2) = 1/2$$

Expected number of $H$’s from one flip of a fair coin is $1/2$.

Q: What is the expected number of Heads in $n$ tosses of a coin?

Let $X = \sum_{i=1}^{n} X_{H,i}$ be the total number of $H$’s in $n$ tosses.

Then:

$$E[X] = E[\sum_{i=1}^{n} X_{H,i}] = \sum_{i=1}^{n} E[X_H] = \sum_{i=1}^{n} 1/2 = n/2$$
Q: What is the expected number of Heads from one coin toss?

Introduce binary random variable $X_H$ to track this number

$$E[X_H] = 1 \cdot P(X_H = 1) + 0 \cdot P(X_H = 0) = 1 \cdot (1/2) + 0 \cdot (1/2) = 1/2$$

*Expected number of H’s from one flip of a fair coin is 1/2.*

Q: What is the expected number of Heads in $n$ tosses of a coin?

Let $X = \sum_{i=1}^{n} X_{H,i}$ be the total number of H’s in $n$ tosses.

Then:

$$E[X] = E[\sum_{i=1}^{n} X_{H,i}] = \sum_{i=1}^{n} E[X_{H}]$$

$$= \sum_{i=1}^{n} 1/2 = n/2$$

*Expected number of H’s from $n$ tosses of a fair coin is 1/2.*
Q: What is the expected number of Heads from one coin toss?

Introduce binary random variable $X_H$ to track this number

$$E[X_H] = 1 \cdot P(X_H = 1) + 0 \cdot P(X_H = 0) = 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{2} = \frac{1}{2}$$

*Expected number of H’s from one flip of a fair coin is 1/2.*

Q: What is the expected number of Heads in $n$ tosses of a coin?

Let $X = \sum_{i=1}^{n} X_{H,i}$ be the total number of H’s in $n$ tosses.

Then:

$$E[X] = E[\sum_{i=1}^{n} X_{H,i}] = \sum_{i=1}^{n} E[X_H]$$

$$= \sum_{i=1}^{n} \frac{1}{2} = \frac{n}{2}$$

*Expected number of H’s from $n$ tosses of a fair coin is 1/2.*
Back to Average Case Analysis of Insertion Sort

**InsertionSort**(*array*\text{A}[1 \ldots n])

1. for \( j \leftarrow 2 \) to \( n \) do
2. \hspace{1em} Temp \leftarrow A[j]
3. \hspace{1em} i \leftarrow j - 1
4. \hspace{1em} while \( i > 0 \) and \( A[i] > \) Temp do
5. \hspace{2em} A[i + 1] \leftarrow A[i]
6. \hspace{2em} i \leftarrow i - 1
7. \hspace{1em} A[i + 1] \leftarrow Temp

- Loop invariant: At the start of each iteration \( j \), \( A[1 \ldots j - 1] \) is sorted.

Recall:
\[ T(n) = \sum_{j=2}^{n} \{A + \sum_{i=0}^{j-1} B + C\} \]

Ignoring machine-dependent constants, we can write:
\[ T(n) = \sum_{j=2}^{n} k_j \], where \( k_j \) is a variable that tracks the total number of iterations of the inner while loop in an iteration of the outer for loop.

In the worst-case analysis, we assumed that \( k_j \leq j \), arriving at a total quadratic running time for insertion sort.

*Here we ask for \( E[k_j] \).*
**Average Case Analysis of Insertion Sort**

$k_j$: random variable counting total number of moves to the right

So: $E[k_j] = E[\sum_{i=1}^{j-1} k_i]$, where $k_i$ is a random variable tracking the number of moves in one iteration of the while loop

By linearity of expectation: $E[k_j] = \sum_{i=1}^{j-1} E[k_i]$

What is $E[k_i]$?
Average Case Analysis of Insertion Sort

- $k_j$: random variable counting total number of moves to the right

So: $E[k_j] = E[\sum_{i=1}^{j-1} k_i]$, where $k_i$ is a random variable tracking the number of moves in one iteration of the while loop

By linearity of expectation: $E[k_j] = \sum_{i=1}^{j-1} E[k_i]$

What is $E[k_i]$?

$E[k_i] = P(move) \times 1 + P(no\ move) \times 0$
Average Case Analysis of Insertion Sort

\( k_j \): random variable counting total number of moves to the right

So: \( E[k_j] = E[\sum_{i=1}^{j-1} k_i] \), where \( k_i \) is a random variable tracking the number of moves in one iteration of the while loop

By linearity of expectation: \( E[k_j] = \sum_{i=1}^{j-1} E[k_i] \)

What is \( E[k_i] \)?

\( E[k_i] = P(\text{move}) \times 1 + P(\text{no move}) \times 0 \)

\( P(\text{move}) = P(A[i] > \text{Key}) = 0.5 \)
Average Case Analysis of Insertion Sort

\( k_j \): random variable counting total number of moves to the right

So: \( E[k_j] = E[\sum_{i=1}^{j-1} k_i] \), where \( k_i \) is a random variable tracking the number of moves in one iteration of the while loop

By linearity of expectation: \( E[k_j] = \sum_{i=1}^{j-1} E[k_i] \)

What is \( E[k_i] \)?

\( E[k_i] = P(\text{move}) \times 1 + P(\text{no move}) \times 0 \)

\( P(\text{move}) = P(A[i] > \text{Key}) = 0.5 \)

So: \( E[k_i] = 0.5 \times 1 = 0.5 \implies E[k_j] = \sum_{i=1}^{j-1} 0.5 = \frac{j-1}{2} \)
Average Case Analysis of Insertion Sort

\( k_j: \) random variable counting total number of moves to the right

So: \( E[k_j] = E[\sum_{i=1}^{j-1} k_i], \) where \( k_i \) is a random variable tracking the number of moves in one iteration of the while loop

By linearity of expectation: \( E[k_j] = \sum_{i=1}^{j-1} E[k_i] \)

What is \( E[k_i]? \quad E[k_i] = P(\text{move}) \times 1 + P(\text{no move}) \times 0 \)

\( P(\text{move}) = P(A[i] > \text{Key}) = 0.5 \)

So: \( E[k_i] = 0.5 \times 1 = 0.5 \quad \implies \quad E[k_j] = \sum_{i=1}^{j-1} 0.5 = \frac{j-1}{2} \)

Finally: \( E[T(n)] = \sum_{j=2}^{n} \frac{j-1}{2} \)
Average Case Analysis of Insertion Sort

$k_j$: random variable counting total number of moves to the right

So: $E[k_j] = E[\sum_{i=1}^{j-1} k_i]$, where $k_i$ is a random variable tracking the number of moves in one iteration of the while loop

By linearity of expectation: $E[k_j] = \sum_{i=1}^{j-1} E[k_i]$

What is $E[k_i]$? $E[k_i] = P(move) \times 1 + P(no\ move) \times 0$

$P(move) = P(A[i] > Key) = 0.5$

So: $E[k_i] = 0.5 \times 1 = 0.5 \implies E[k_j] = \sum_{i=1}^{j-1} 0.5 = \frac{j-1}{2}$

Finally: $E[T(n)] = \sum_{j=2}^{n} \frac{j-1}{2}$

You can show that this expected running time is no better than the worst-case running time.
Average Case Analysis of Insertion Sort

$k_j$: random variable counting total number of moves to the right

So: $E[k_j] = E[\sum_{i=1}^{j-1} k_i]$, where $k_i$ is a random variable tracking the number of moves in one iteration of the while loop

By linearity of expectation: $E[k_j] = \sum_{i=1}^{j-1} E[k_i]$

What is $E[k_i]$? $E[k_i] = P(\text{move}) \times 1 + P(\text{no move}) \times 0$

$P(\text{move}) = P(A[i] > \text{Key}) = 0.5$

So: $E[k_i] = 0.5 \times 1 = 0.5 \implies E[k_j] = \sum_{i=1}^{j-1} 0.5 = \frac{i-1}{2}$

Finally: $E[T(n)] = \sum_{j=2}^{n} \frac{j-1}{2}$

You can show that this expected running time is no better than the worst-case running time.
Can we do better than $\theta(n^2)$?

You have already seen an example ...
Can we do better than $\theta(n^2)$?

You have already seen an example ...

More follow
Can we do better than $\theta(n^2)$?

You have already seen an example ...

More follow
1 Outline of Today’s Class
   • Sorting in $O(n \lg n)$ Time on Average: Quicksort
Quicksort: Divide and Conquer

- Proposed by C. A. R. Hoare in 1962
- Implements the divide-and-conquer paradigm
- Is a very practical algorithm
- Sorts in place like insertion sort and heapsort
  1. Divide: Partition array into two subarrays around a pivot \( x \) s.t. values left \( \leq x \leq \) values right
  2. Conquer: Recursively sort the two subarrays
  3. Combine: Trivial

Key to speed: linear-time partitioning subroutine
PARTITION(A, p, q)

1: \( x \leftarrow A[p] \)
2: \( i \leftarrow p \)
3: for \( j \leftarrow p + 1 \) to \( q \) do
4: \( \text{if } A[j] \leq x \text{ then} \)
5: \( i \leftarrow i + 1 \)
6: \( \text{swap}(A[i], A[j]) \)
7: return \( i \)

Running time = \( O(n) \) for \( n \) elements.
Partitioning: Trace

\[ 6 \quad 10 \quad 13 \quad 5 \quad 8 \quad 3 \quad 2 \quad 11 \]

\[ i \quad j \]
Partitioning: Trace

\[
\begin{array}{cccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
\end{array}
\]

\[i \quad j\]

\[
\begin{array}{cccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
\end{array}
\]

\[i \quad \rightarrow \quad j\]
### Partitioning: Trace

| 6 | 10 | 13 | 5 | 8 | 3 | 2 | 11 |

\[ i \quad j \]

| 6 | 10 | 13 | 5 | 8 | 3 | 2 | 11 |

\[ i \quad j \]

| 6 | 10 | 13 | 5 | 8 | 3 | 2 | 11 |

\[ i \quad j \]
Outline of Today’s Class

Sorting in \(O(n \log n)\) Time on Average: Quicksort

Partitioning: Trace

\[
\begin{array}{cccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
\end{array}
\]

\(i\) \quad \(j\)

\[
\begin{array}{cccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
\end{array}
\]

\(i\) \quad \rightarrow \quad j

\[
\begin{array}{cccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
\end{array}
\]

\(i\) \quad \rightarrow \quad j

\[
\begin{array}{cccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
6 & 5 & 13 & 10 & 8 & 3 & 2 & 11 \\
\end{array}
\]

\rightarrow \quad i \quad \quad j
Outline of Today’s Class

Sorting in $O(n \lg n)$ Time on Average: Quicksort

Partitioning: Trace

```
6 10 13 5 8 3 2 11
```

- $i \quad j$

```
6 10 | 13 5 8 3 2 11
```

- $i \quad j$

```
6 10 13 | 5 8 3 2 11
```

- $i \quad j$

```
6 5 | 13 10 8 3 2 11
```

- $i \quad j$
Partitioning: Trace

6 10 13 5 8 3 2 11

6 5 13 10 8 3 2 11

i j

Outline of Today’s Class
Sorting in $O(n \log n)$ Time on Average: Quicksort
Outline of Today’s Class

Sorting in $O(n \log n)$ Time on Average: Quicksort

Partitioning: Trace

$$\begin{array}{cccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
\end{array}$$

$$\begin{array}{cccccccc}
6 & 5 & 13 & 10 & 8 & 3 & 2 & 11 \\
\end{array}$$

$$\begin{array}{cccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
\end{array}$$

$$\begin{array}{cccccccc}
6 & 5 & 13 & 10 & 8 & 3 & 2 & 11 \\
\end{array}$$
Partitioning: Trace
Partitioning: Trace

6 10 13 5 8 3 2 11

6 5 13 10 8 3 2 11

6 5 3 10 8 13 2 11

\[ \rightarrow \quad i \quad \rightarrow j \]
Partitioning: Trace
Outline of Today’s Class

Sorting in $O(n \log n)$ Time on Average: Quicksort

Partitioning: Trace

1. Original array: 6 10 13 5 8 3 2 11
2. First partition: 6 5 13 10 8 3 2 11
3. Second partition: 6 5 3 10 8 13 2 11
4. Partitioning: $i \rightarrow j$
5. Final array: 6 10 13 5 8 3 2 11
6. Final partition: 6 5 13 10 8 3 2 11
7. Final partition: 6 5 3 10 8 13 2 11
8. Partitioning: $i \rightarrow j$
Partitioning: Trace

6 10 13 5 8 3 2 11

6 5 13 10 8 3 2 11

6 5 3 10 8 13 2 11

6 5 3 2 8 13 10 11

i  j
Partitioning: Trace

\[ 6 \ 10 \ 13 \ 5 \ 8 \ 3 \ 2 \ 11 \]

\[ 6 \ 5 \ 13 \ 10 \ 8 \ 3 \ 2 \ 11 \]

\[ 6 \ 5 \ 3 \ 10 \ 8 \ 13 \ 2 \ 11 \]

\[ 6 \ 5 \ 3 \ 2 \ 8 \ 13 \ 10 \ 11 \]

\( i \) \quad \rightarrow \quad \( j \)
Partitioning: Trace

\[ \begin{array}{ccccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
\hline
6 & 5 & 13 & 10 & 8 & 3 & 2 & 11 \\
\hline
6 & 5 & 3 & 10 & 8 & 13 & 2 & 11 \\
\hline
6 & 5 & 3 & 2 & 8 & 13 & 10 & 11 \\
\end{array} \]

\(i\) \[\rightarrow\] \(j\)
QuickSort: Pseudocode And Analysis

QuickSort(A, p, r)

1: if p < r then
2:    q ← PARTITION(A, p, r)
3:    QUICKSORT(A, p, q − 1)
4:    QUICKSORT(A, q + 1, r)

Initial call: QUICKSORT(A, 1, n)

Worst-case Time Analysis:
- Assume elements are distinct
- There are better algorithms for duplicate elements

- Let $T(n)$ be worst-case running time on $n$ elements
- $A$ is sorted/reverse sorted; partition around min/max element
- One side of partition always has no elements

\[
T(n) = T(0) + T(n - 1) + \theta(n)
\]
\[
= \theta(1) + T(n - 1) + \theta(n)
\]
\[
= T(n - 1) + \theta(n) - \text{arithmetic series}
\]
\[
= \theta(n^2)
\]
Worst-case Recursion Tree

\[ T(n) = T(0) + T(n-1) + cn \]

\[ \Theta \left( \sum_{k=1}^{n} k \right) = \Theta (n^2) \]

\[ T(n) = \Theta(n) + \Theta(n^2) = \Theta(n^2) \]
Best Case:

- If we are lucky, PARTITION splits the array evenly
- \[ T(n) = 2T(n/2) + \theta(n) = \theta(n \log n) \]
- Let \( L(n) \) denote the running time when we are lucky
- Versus \( U(n) \) - the worst-case running time of \( \theta(n^2) \)

Almost Best Case:

- What if the split is not even?
- Say, it is \( \frac{1}{10} : \frac{9}{10} \)
- \[ T(n) = T(\frac{1}{10} n) + T(\frac{9}{10} n) + \theta(n) \]
- What is the solution to this recurrence?
Analysis of "almost best"

\[ \log_{10} n \]

\[ \frac{1}{100} \]

\[ cn \]

\[ \frac{9}{10} \]

\[ cn \]

\[ \frac{9}{100} \]

\[ cn \]

\[ \frac{81}{100} \]

\[ cn \]

\[ \Theta(1) \]

\[ O(n) \text{ leaves} \]

\[ \Theta(n \log n) \]

Lucky!

\[ cn \log_{10} n \leq T(n) \leq cn \log_{10/9} n + O(n) \]
More Intuition

- Suppose that QUICKSORT is alternately lucky, unlucky, lucky, unlucky, lucky, ...
  \[ L(n) = 2U(n/2) + \theta(n) \]
  \[ U(n) = L(n - 1) + \theta(n) \]

- Solving further:
  \[ L(n) = 2(L(n/2 - 1/2) + \theta(n/2)) + \theta(n) \]
  \[ = 2L(n/2 - 1/2) + \theta(n) \]
  \[ = \theta(n\log n) - \text{Lucky!!!} \]

- How can we make sure QUICKSORT is *usually* lucky?
Randomized Quicksort

**Basic Idea:** Partition around a *random* element

- Running time is independent of input order
- No assumptions need to be made about the input distribution
- No specific input elicits the worst-case behavior
- The worst case is determined now only by the output of a random-number generator
Randomized Quicksort Analysis

- Let $T(n)$ be the random variable for the running time of randomized quicksort on an input of length $n$, assuming random numbers are independent.

- So:

$$T(n) = \begin{cases} 
T(0) + T(n-1) + \theta(n) & \text{if } 0:n-1 \text{ split} \\
T(1) + T(n-2) + \theta(n) & \text{if } 1:n-2 \text{ split} \\
\ldots & \\
T(n-1) + T(0) + \theta(n) & \text{if } n-1:0 \text{ split}
\end{cases}$$

- Each of these $k : n-k-1$ partitions ($k \in \{0, 1, \ldots, n-1\}$ is equally likely, assuming distinct elements).

- So: $E[T(n)] = \frac{1}{n} \sum_{k=0}^{n-1} \{ E[T(k)] + E[T(n-k-1)] + \theta(n) \}$
Randomized Quicksort Analysis Continued

Continuing:

\[ E[T(n)] = \frac{1}{n} \sum_{k=0}^{n-1} \{ E[T(k)] + E[T(n-k-1)] + \theta(n) \} \]
\[ = \frac{1}{n} \sum_{k=0}^{n-1} \{ E[T(k)] + E[T(n-k-1)] \} + \frac{1}{n} \sum_{k=0}^{n-1} \theta(n) \]
\[ = \frac{1}{n} \sum_{k=0}^{n-1} \{ E[T(k)] + E[T(n-k-1)] \} + \frac{1}{n} \cdot n \cdot \theta(n) \]
\[ = \frac{1}{n} \sum_{k=0}^{n-1} \{ E[T(k)] + \theta(n) \} \]
\[ = \frac{1}{n} \sum_{k=0}^{n-1} E[T(k)] + \frac{1}{n} \sum_{k=0}^{n-1} E[T(n-k-1)] + \theta(n) \]

Summations have identical terms

\[ = \frac{2}{n} \sum_{k=0}^{n-1} E[T(k)] + \theta(n) \]

What do we do now?
Randomized Quicksort Analysis Continued

- The $k = 0, 1$ terms can be absorbed in the $\theta(n)$
- So: $E[T(n)] = \frac{2}{n} \sum_{k=2}^{n-1} \{E[T(k)]\} + \theta(n)$

**Guess:** $E[T(n)] \in O(n \lg n)$

By induction, need to find $a > 0$ s.t. $E[T(n)] \leq a \cdot n \cdot \lg n$

Use the fact that $\sum_{k=2}^{n-1} k \cdot \lg k \leq \frac{1}{2} n^2 \cdot \lg n - \frac{1}{4} n^2$ (integration technique bounds this summation)

Then, using the substitution/induction technique:

$$E[T(n)] \leq \frac{2}{n} \sum_{k=2}^{n-1} a \cdot k \cdot \lg k + \theta(n)$$

$$= \frac{2a}{n} \left( \frac{1}{2} n^2 \cdot \lg n - \frac{1}{4} n^2 \right) + \theta(n)$$

$$= a \cdot n \cdot \lg n - \left( \frac{an}{2} - \theta(n) \right)$$

$$\leq a \cdot n \cdot \lg n$$

Note: $a$ needs to be large enough so that $\frac{an}{2}$ dominates $\theta(n)$
Final Word on Quicksort

- Useful general-purpose algorithm
- Typically over twice as fast as mergesort
- Can benefit substantially from code tuning
- Behaves well even with caching and virtual memory