Agenda

1. Announcements
2. Review: Proability Rules
3. Homework 7 Questions
4. Time Complexity
5. Recurrence Relations
6. Divide and Conquer/Master Theorem

iClicker Frequency: BA
Announcements

- ABK – Regrades/Makeup Grades Posted
- ABK – Survey sufficiently filled in, dropping a quiz
- Rob will be at Home Plate 8-10 tonight.
- I will have Porter’s Pub hour Friday 2-4 pm (followed by Pepband performance)
- Other office hours are often emptyish, please come by if you need help
- Homework 8 due **Tuesday** 5/27 at midnight

iClicker Frequency: BA
Marginalization:

\[ P(a) = \sum_{b \in B} P(a, b) \]

Conditional Probability:

\[ P(a \mid b) = \frac{P(a, b)}{P(b)} \]

Product Rule:

\[ P(a, b, c) = P(a)P(b \mid a)P(c \mid a, b) \]

Bayes Rule:

\[ P(a \mid b) = \frac{P(b \mid a)P(a)}{P(b)} \]
Question: A test is designed to detect a disease. Among those who have the disease, the probability that the disease will be detected by the new test is 0.74. However, the probability that the test will erroneously indicate the presence of the disease in those who do not actually have it is 0.04. It is estimated that 16% of the population who take this test have the disease. If the test administered to an individual is positive, what is the probability that the person actually has the disease?

Solution:

- Set up model $D = $ Diseased, $P = $ Positive test
- Interpret the story $P(P|D) = 0.74$, $P(P|D') = 0.04$, $P(D) = 0.16$
- Interpret goal $P(D|P)$
**Solution:**

- Set up model $D =$ Diseased, $P =$ Positive test
- Interpret the story $P(P|D) = 0.74, \ P(P|D') = 0.04, \ P(D) = 0.16$
- Interpret goal $P(D|P)$
- Apply Bayes Rule $P(D|P) = \frac{P(P|D)P(D)}{P(P)}$
- Marginalize to get
  
  $P(P) = P(P, D) + P(P, D') = P(D)P(P|D) + P(D')P(P|D')$

- $P(D|P) = \frac{P(P|D)P(D)}{P(D)P(P|D) + P(D')P(P|D')}$
- $P(D|P) = \frac{0.74 \times 0.16}{0.16 \times 0.74 + (1 - 0.16) \times 0.04} \approx 0.77$
Question: You ask a neighbor to water a sickly plant while you are on vacation. Without water the plant will die with probability 0.8. With water it will die with probability 0.5. You are 89% certain the neighbor will remember to water the plant. When you are on vacation, find the probability that the plant will die.

Solution:

- Set up model $D =$ Plat Dies, $W =$ Plant Watered
- Interpret the story $P(D|W') = 0.8$, $P(D|W) = 0.5$, $P(W) = 0.89$
- Interpret goal $P(D)$
- Apply marginalization $P(D) = P(D, W) + P(D, W')$
- Product rule $P(D) = P(W)P(D|W) + P(W')P(D|W')$
- Substitute $P(D) = 0.89 \times 0.5 + (1 - 0.89) \times 0.8 \approx 0.53$
Given $P(D|W') = 0.8$, $P(D|W) = 0.5$, $P(W) = 0.89$, and $P(D) = 0.53$, what is $P(W|D)$?

(a) $\frac{0.8 \times 0.89}{0.53}$

(b) $\frac{0.5 \times 0.89}{0.53}$

(c) $\frac{0.8 \times 0.53}{0.89}$

(d) $\frac{0.5 \times 0.53}{0.89}$

(e) None of the above
Time complexity establishes a bound on how many operations an algorithm will take as a function of input. Example: Summing a list of $n$ elements requires $n - 1$ additions in the standard algorithm.

Key observations:
- What an “operation” is depends on the type of algorithm/level of abstraction
  - Matrix multiply – single multiply
  - Graph search – single visit to a node
  - Learning algorithms – operating on a record
- You really don’t care exactly how many operations, just an order of magnitude (since each operation can take a constant multiple of time)
- Related topic is “memory complexity” where memory usage is measured in a similar manner
Time Complexity

Algorithm A:

\[
\text{for } i \leftarrow 1 \text{ to } n:\n\quad y += i \\
\quad z -= i
\]

Algorithm B:

\[
\text{for } i \leftarrow 1 \text{ to } n:\n\quad y += i \\
\quad \text{for } j \leftarrow 1 \text{ to } n:\n\quad z -= j
\]

iClicker Question (Frequency: BA)

Which algorithm has bigger time complexity?

(a) \( A > B \)

(b) \( A = B \)  

(c) \( A < B \)

(d) Impossible to know
Algorithm A:

\[
\text{for } i \leftarrow 1 \text{ to } n: \\
x += i
\]

Algorithm B:

\[
\text{for } i \leftarrow 1 \text{ to } m: \\
x += 1 \\
y += i \\
z -= i
\]

iClicker Question (Frequency: BA)

Which algorithm has bigger time complexity?

(a) \( A > B \)
(b) \( A = B \)*
(c) \( A < B \)
(d) Impossible to know
Time Complexity

Algorithm A:

```
for i <- 1 to n:
  x += i
```

Algorithm B:

```
for i <- 1 to m:
  for j <- 1 to n:
    y += i
    z -= j
```

iClicker Question (Frequency: BA)

Which algorithm has bigger time complexity?

(a) $A > B$
(b) $A = B$
(c) $A < B$
(d) Impossible to know
Asymptotic Bounds

Time complexity is typically measured in *asymptotic bounds*: As $N$ goes to $\infty$, the function never crosses the bound

- **$\text{Big } O$** $O(f(n))$ – Upper bound
- **$\text{Big } \Omega$** $\Omega(f(n))$ – Lower bound
- **$\text{Big } \Theta$** $\Theta(f(n))$ – Upper & Lower bound

Reduce to fewest terms

- Number of operations upto a constant factor $2f(n) = \Theta(f(n))$
- Throw away all but largest term $n^2 + n + 2 = \Theta(n^2)$
- If there are multiple variables, typically keep largest term of each variable $n^2 + mn + m^2 = O(n^2 + m^2)$
\( x^2 \) and \( x^2 + 50x \)
$x^2$ and $x^2 + 50x$
\(x^2\) and \(x^2 + 50x - x^2 + 50x = \Theta(x^2)\)
Here is a list of common Big O classes

1. Constant \( O(1) \)
2. Logarithmic \( O(\log(n)) \)
3. Linear \( O(n) \)
4. \( O(n \log(n)) \) – Sorting/Binary Search
5. Quadratic \( O(n^2) \)
6. Polynomial \( O(n^k) \) for \( k \geq 1 \)
7. **Below here is untenable for reasonable inputs**
6. Exponential \( O(a^n) \)
9. Factorial \( O(n!) \)
for i <- 1 to n:
    for j <- i to m:
        x+=1
for i <- 1 to n:
    x += 2

iClicker Question (Frequency: BA)

Which of the following is the most reduced bound for this code?

(a) $\Theta(n)$ or $\Theta(m)$
(b) $\Theta(n^2 + n)$
(c) $\Theta(nm + n)$
(d) $\Theta(nm)$
(e) $\Theta(n^2)$ *
A recurrence relation defines a sequence as a function of previous elements. **Example:** fibonacci sequence:  \( F_n = F_{n-1} + F_{n-2} \) 0,1,1,2,3,5,...  These are difficult to reason about directly so you typically want to solve the recurrence into a non recursive form.  There are several methods:  
- Induction  
- Unfolding  
- Characteristic Equation  
- Master Theorem
Induction is a mathematical axiom that is used to prove a statement by showing:

(a) The statement holds at the start of the sequence
(b) The statement holding for $n$ implies it holds for $n + 1$

Use this template for inductive proofs:

**Base case**  Show that the statement holds for some initial case (usually $n = 1$ or 0)

**Inductive hypothesis**  Postulate that $f(n) \Rightarrow f(n + 1)$

**Inductive step**  show $f(n) \Rightarrow f(n + 1)$

**Conclusion**  “Hence by induction $f(n)$ holds for all cases”
Sum of first $n$ integers: $s(n) = s(n - 1) + n$, $s(1) = 1$

Prove this is a solution: $s(n) = \frac{n(n+1)}{2}$

**Base Case:**
$s(1) = 1 = \frac{1(1+1)}{2}$

**Inductive Hypothesis**
$s(n) = \frac{n(n+1)}{2} \implies s(n + 1) = \frac{(n+1)(n+2)}{2}$

**Inductive Step**
Assume $s(n) = \frac{n(n+1)}{2}$.

Then $s(n + 1) = \frac{n(n+1)}{2} + n + 1$.

Then $s(n + 1) = (n + 1) \left( \frac{n}{2} + 1 \right)$

Then $s(n + 1) = (n + 1) \left( \frac{n+2}{2} \right)$

Then $s(n + 1) = \frac{(n+1)((n+1)+1)}{2}$, as required.

Hence by induction $s(n) = \frac{n(n+1)}{2}$.
Recurrence Relation – Unfolding

Substitute a few elements in until a pattern emerges, then figure out that pattern

- \( g(n) = g(n - 1) + 2n - 1, \quad g(0) = 0 \)
- \( = g(n - 2) + 2(n - 1) - 1 + 2n - 1 \)
- \( = g(n - 3) + 2(n - 2) + 2(n - 1) + 2n - 3 \)
- \( = g(n - i) + 2(n - i + 1) + ... + 2n - i \)
- \( = g(n - n) + 2(n - n + 1) + 2(n - (n - 1) + 1) + ... + 2n - n \)
- \( = 0 + 2 + 4 + ... + 2n - n \)
- \( = 2 (\sum_{i=0}^{n} i) - n \)
- \( = 2 \frac{n(n+1)}{2} - n \)
- \( = n^2 \)
If a recurrence can be written in the form:

\[ a_n = ba_{n-1} + ca_{n-2}, \quad a_0 = i \quad a_1 = j \]

Then it can be solved via a Characteristic Equation.

\[ x^2 - bx - c = 0 \]

Let \( r_1 \) and \( r_2 \) be the roots of the polynomial, then there are two cases:

1. \( r_1 \neq r_2 \)
   - Solution: \( a_n = K_1 r_1^n + K_2 r_2^n \), such that:
     - \( K_1 + K_2 = a_0 \)
     - \( r_1 K_1 + r_2 K_2 = a_1 \)

2. \( r_1 = r_2 \)
   - Solution: \( a_n = K_1 r_1^n + K_2 nr_1^n \), such that:
     - \( K_1 = a_0 \)
     - \( r_1 K_1 + r_1 K_2 = a_1 \)
Gambler’s Ruin

A gambler starts with $k$ dollars and repeatedly bets $1$ that a coin will come up heads. Thus, the gambler earns $1$ if heads and loses $1$ if tails. If the gambler reaches $M$ dollars, he’ll stop ($M > k$). What is the probability the gambler will lose all his money?

- Let $P_k$ be the probability of losing $k$ dollars, $x$ is next coin flip, $R$ is eventual ruin
- $P_k = P(x = h, R) + P(x = t, R) = P(h)P(R|h) + P(t)P(R|t)$
- Since $P(h) = P(t) = \frac{1}{2} \rightarrow 0.5P(R|h) + 0.5P(R|t)$
- Now becomes a recurrence: $P_k = 0.5P_{k+1} + 0.5P_{k-1}$
- Rearrange: $-0.5P_{k+1} = -P_k + .5P_{k-1} \rightarrow P_{k+1} = 2P_k - P_{k-1}$
- Redefine $a = k + 1$, $P_a = 2P_{a-1} - P_{a-2}$
We just learned \( P_a = 2P_{a-1} - P_{a-2} \).
What is the characteristic equation of this recurrence?

(a) \( r^2 - 2r \)
(b) \( -2r^2 + r \)
(c) \( r^2 + r - 2 \)
(d) \( r^2 - 2r + 1 \)
Gambler’s Ruin

We now solve our characteristic equation:

\[ r^2 - 2r + 1 = 0 \]

\[ (r - 1)(r - 1) = 0 \]

We have repeated roots \( r = 1 \).

We know \( P_0 = 1 \) and \( P_M = 0 \) since certain ruin if 0 and certain win if \( M \).

With repeated roots, we get:

\[ a_n = K_1 r^n + K_2 nr^n \]
\[ a_0 = K_1 \quad a_1 = rK_1 + rK_2 \]

Plugging in and solving:

\[ P_M = K_1 r^M + K_2 Mr^M = P_0 + K_2 M \rightarrow K_2 = -\frac{1}{M} \]

Final substitution:

\[ P_n = 1 - \frac{n}{M} \]
Divide and Conquer

Divide and conquer is an algorithms principle where a big problem is broken down into several smaller sub problems. Their complexity is governed by a recurrence relation.

**Example: Merge Sort**

- Given a size $n$ list ($L$), sort it
- Break it into two lists, $L[0 : n/2]$ and $L[n/2 : n]$
- Recursively sort the two lists using merge sort
- Merge the two lists, so that they are sorted (see board)
- The time to do this is $T(n) = 2 \times T(n/2) + n$
Mere sort splits data into two at each level; however, total work per recursive level is still $\Theta(n)$

Furthermore there are $\log_2(n)$ levels since repeatedly split into 2 nodes. Thus the whole algorithm is $O(n \log(n))$
The Master Theorem says that given a recurrence like:

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^c) \quad T(1) = d$$

The solution is given by one of three case:

1. if $\log_b a < c$, $T(n) = \Theta(n^c)$
2. if $\log_b a = c$, $T(n) = \Theta(n^c \log n)$
3. if $\log_b a > c$, $T(n) = \Theta(n^{\log_b a})$

This allows easy solving of certain DQ algorithm complexities.
Solve this complexity via the master theorem.

\[ T(n) = 10T\left(\frac{n}{5}\right) + \Theta(n) \]

(a) \( T(n) = \Theta(n) \)
(b) \( T(n) = \Theta(n \log n) \)
(c) \( T(n) = \Theta(n^{\log_5 10}) \)
(d) None of the above – cannot use master theorem
Solve this complexity via the master theorem.

\[ T(n) = 2T\left(\frac{n}{4}\right) + \Theta(n^2) \]

(a) \( T(n) = \Theta(n^2) \)
(b) \( T(n) = \Theta(n^2 \log n) \)
(c) \( T(n) = \Theta(n^{\log_4 2}) \)
(d) None of the above – cannot use master theorem
Solve this complexity via the master theorem.

$$T(n) = 2T(n - 1) + \Theta(n)$$

(a) $T(n) = \Theta(n)$
(b) $T(n) = \Theta(n \log n)$
(c) $T(n) = \Theta(n^{\log_2 2})$
(d) None of the above – cannot use master theorem*
iClicker Question (Frequency: BA)

Solve this complexity via the master theorem.

\[ T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n^2) \]

(a) \( T(n) = \Theta(n^2) \)
(b) \( T(n) = \Theta(n^2 \log n) \)
(c) \( T(n) = \Theta(n^{\log_2 4}) \)
(d) None of the above – cannot use master theorem