Lecture 13 Notes

• Goals for this week
  – (Tuesday) Conditional probability and Bayes’ rule
  – Induction and recursion, on to complexity (GT Sec. 4)!
  – For quiz: Tuesday material

• Feedback survey (anonymous google doc)
  – ~80 responses so far are very helpful – if not immediately
    then in future courses (e.g., quizzes, WeBWorK, etc.)
  – Please fill out by tonight - not yet at 80% response rate!

• “Midterm consolidation”
  – Sunday, May 18, 6pm in CSE building Rooms 1202, 2154, 4140 – you must be there in person (will not be possible
to take remotely)
  – Note: this cannot hurt your grade
    • This is true even for students with >90 scores, since any fix to
      ‘miscalibration’ (e.g., on final exam) would just shift scores upward

• No W, F OHs this week (travel):Th 12:30-1:30 + appt
Main Points From Last Time

- Conditional probability facts
  - \( P(B \mid A) = \frac{P(A \cap B)}{P(A)} \) if \( P(A) \neq 0 \), undefined if \( P(A) = 0 \)
  - \( P(B \mid U) = P(B) \)
  - \( A \) and \( B \) are independent iff \( P(B \mid A) = P(B) \)

- Bayes’ Rule: \( P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)} \)
  \[= \frac{P(B \mid A) \cdot P(A)}{[P(B \mid A) \cdot P(A) + P(B \mid A^c) \cdot P(A^c)]} \]

- Divide and conquer
  - If problem is small enough, solve it
  - Else:
    - Decompose into smaller subproblems (“divide”)
    - Recursively solve the subproblems (“conquer”)
    - Recombine subproblem solutions into problem solution (“compose”)
  - Example: Mergesort
• **Q1:** Events A and B are independent. $P(A) = 0.6$ and $P(B) = 0.2$. What is $P(A \mid B)$?

   - **A:** 0.6
   - **B:** $0.6 \cdot 0.2 = 0.12$
   - **C:** 0.2
   - **D:** None of the above

• A drug test detects PED use 80% of the time. However, 5% of PED-free individuals also test positive (= false-positive rate). 10% of NFL players use PEDs. Your NFL player friend tests positive. Recall:

   \[
P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)} = \frac{P(B \mid A) \cdot P(A)}{P(B \mid A) \cdot P(A) + P(B \mid A^c) \cdot P(A^c)}
   \]

• **Q2:** Given that your NFL player friend tests positive for PEDs, what is the probability that he actually used PEDs?

   - **A:** Around 2/3
   - **B:** Around 1/3
   - **C:** Around 9/10

• **Q3:** The number of operations needed to merge two sorted lists of M and N elements, respectively, into a single sorted list is approximately:

   - **A:** $\max(M, N)$
   - **B:** $\min(M, N)$
   - **C:** $M \cdot N$
   - **D:** $M + N$

Why not $2 \times \min$?
- Small list has the large #s
- Output size is a LB on runtime
DQ: Mergesort

• Sorting a list of numbers
  – Divide (into two equal parts)
  – Conquer (solve for each part separately)
  – Combine separate solutions

If list has length one, it’s already sorted (return list)
Else
  Sort first half of list (recursively) \( M \left( \frac{n}{2} \right) \)
  Sort second half of list (recursively) \( M \left( \frac{n}{2} \right) \)
  Merge the two sorted lists \( n \)
Merging Two Sorted Subsequences

Sequence #1: \( x[1], x[2], \ldots, x[m] \)
Sequence #2: \( y[1], y[2], \ldots, y[n] \)

if \( y[i] < x[j] \) then compare \( y[i+1] \) and \( x[j] \),
else compare \( y[i] \) and \( x[j+1] \)

After comparing \( x[1] \) and \( y[1] \), advance in \( x[.] \) and \( y[.] \)

total of \((m-1) + (n-1)\) times \(\Rightarrow\) \( m + n - 1 \) comparisons
(edges between red and blue) = linear time
Tree of Recursions in Mergesort

Check out: https://www.youtube.com/watch?v=XaqR3G_NVoo

- n comparisons per level
- log n levels
- total runtime looks to be around \( n \cdot \log n \)
More Precisely …

- Let $M(n)$ denote Mergesort runtime on list of $n$ numbers

- $M(n) = 1$ if $n = 1$
  
  $= M(n/2) + M(n/2) + n$ if $n > 1$

- Claim: $M(n) = n \cdot \log_2 n + n \quad \forall n \geq 1$

How can we prove this?
Mathematical Induction

- **M(n)** = Mergesort runtime on list of *n* numbers
- **M(n)** = 1 if *n* = 1
- M(n) = M(n/2) + M(n/2) + *n* if *n* > 1
- **Claim:** M(n) = *n* \cdot \log_2 n + *n* \quad \forall n \geq 1
- **Base case:** M(1) = 1 \cdot \log_2 1 + 1 = 1 \cdot 0 + 1 = 1
- **Induction step:** Let *n* be a fixed integer and suppose
  - M(k) = *k* \cdot \log_2 k + *k* for all k < *n*. We want to show that M(n) = *n* \cdot \log_2 n + *n*.
  - M(n) = 2 \cdot M(n/2) + *n*
  - = 2 \cdot [(n/2) \cdot \log_2 (n/2) + (n/2)] + *n* by I.H.
  - M(n/2) = \frac{n}{2} \cdot \log_2 \left(\frac{n}{2}\right) + \frac{n}{2}

\*Side note: review or get comfortable with logs

\*log_a b = \log a \div \log b

\*log_2 (4^n) = \log_2 n + 2

\*\log_2 n - 1
Mathematical Induction

- \( M(n) = \text{Mergesort runtime on list of } n \text{ numbers} \)
- \( M(n) = 1 \quad \text{if } n = 1 \)
- \( = M(n/2) + M(n/2) + n \quad \text{if } n > 1 \)

- **Claim:** \( M(n) = n \cdot \log_2 n + n \quad \forall n \geq 1 \)

- **Base case:** \( M(1) = 1 \cdot \log_2 1 + 1 = 1 \cdot 0 + 1 = 1 \)

- **Induction step:** Let \( n \) be a fixed integer and suppose \( M(k) = k \cdot \log_2 k + k \) for all \( k < n \). We want to show that \( M(n) = n \cdot \log_2 n + n \).

\[
M(n) = 2 \cdot M(n/2) + n
\]

\[
= 2 \cdot [(n/2) \cdot \log_2 (n/2) + (n/2)] + n \quad \text{by I.H.}
\]

\[
= n(\log_2 n - \log_2 2) + n + n = n \cdot \log_2 n - n + n + n
\]

\[= n \cdot \log_2 n + n \quad \checkmark
\]
Another Recursion: Tower of Hanoi

- One disk moves at a time
- Never put larger disk onto smaller disk

What is **minimum** #moves needed to transfer a stack of \( n \) disks?

**Notation**
- For a stack of \( n \) disks, call this number \( T_n \)

**Small Cases**
- \( T_0 = 0 \), \( T_1 = 1 \), \( T_2 = 3 \), \( T_3 = 7 \), ...
TOWER OF HANOI (CONT.)

- Can we “reduce to a known problem”?
  - Note that this is the idea of recursion!
  - \( T_n \leq 2T_{n-1} + 1 \), \( n > 0 \)
  - Why?
    - Shift \((n-1)\), move largest disk, shift again
  - Why the \( \leq \) inequality? = an **upper** bound
    - \( 2T_{n-1} + 1 \) suffices, but maybe can do better
Tower of Hanoi (cont.)

• Can we “reduce to a known problem”?
  – Note that this is the idea of recursion!
  – \( T_n \leq 2T_{n-1} + 1 \), \( n > 0 \)
  – Why?
    Shift \( (n-1) \), move largest disk, shift again
  – Why the \( \leq \) inequality? = an upper bound
    \( 2T_{n-1} + 1 \) suffices, but maybe can do better

• Why does a lower bound \( T_n \geq 2T_{n-1} + 1 \) hold?

\[
\begin{align*}
&\Rightarrow T_n \equiv 2T_{n-1} + 1, \ T_0 = 0 \\
&\text{Comment:} \quad \bullet \ \text{prove} = : \ \text{prove} \leq, \geq \\
&\quad \bullet \ \text{prove} \equiv : \ \text{prove} \Rightarrow, \Leftarrow \\
&\quad \bullet \ \text{prove} = (\text{sets}) : \ \text{prove} \leq, \geq
\end{align*}
\]

• What is a general solution?
Tower of Hanoi (cont.)

• General solution looks like … \( T_n = 2^n - 1 \)
• Let’s **guess** this answer and try to prove it

• **Claim:** \( T_n = 2^n - 1 \)
• **Proof:** (by mathematical induction)
  
  Base case (\( n = 0 \)): \( T_0 = 0 = 2^0 - 1 \) holds
  
  I.H.: \( T_k = 2^k - 1 \) \( \forall k = 0, 1, ..., n - 1 \)
  
  I.S.: \( T_n = 2T_{n-1} + 1 = 2(2^{n-1} - 1) + 1 = 2^n - 1 \) (I.H.)
Another Recursion: Matrix Multiplication

- $A = \ldots$, $B = \ldots$ are $n \times n$ matrices
- $a_{11}, a_{12}, \text{etc.}$ are $n/2 \times n/2$ submatrices
- $M = A \cdot B = \ldots$
  - where $m_{11} = a_{11}b_{11} + a_{12}b_{21}$ etc.
  - Evaluation requires 8 multiplies, 4 adds
- Divide and Conquer: $T(n) = 8T(n/2) + n^2$
Another Recursion: Matrix Multiplication

\[
\begin{pmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{pmatrix}
\times
\begin{pmatrix}
b_{11} & b_{12} \\
b_{21} & b_{22}
\end{pmatrix}
=
\begin{pmatrix}
m_{11} & m_{12} \\
m_{21} & m_{22}
\end{pmatrix}
\]
Another Recursion: Long Multiplication

- Multiplying Large Integers
  \[ A = r^{n-1}a_{n-1} + \ldots + r^1a_1 + a_0, \quad r = \text{radix} \]

- Method that you learned 10 years ago:
  \[
  \begin{array}{c}
  a_{n-1}a_{n-2}\ldots a_1a_0 \\
  \times \quad b_{n-1}b_{n-2}\ldots b_1b_0 \\
  \end{array}
  \]

- What would “Divide and Conquer” look like?
Another Recursion: Long Multiplication
Another Recursion: Determinant

• det (2 × 2 matrix) = ad – bc

• Recursively defined det(M) = (…)
  – $M_{1j}$ is the (1, j) cofactor matrix (or, minor) of $n \times n$ matrix $M$

• det(M) = $m_{11} \cdot \det(M_{11}) + \ldots + (-1)^{1+j}m_{1j} \cdot \det(M_{1j}) + \ldots$

• With this recursive definition, need $\sim n!$ multiplications to calculate the determinant of an $n \times n$ matrix

• What would be a “better method”? 

Basketball Before You Were Born

- No 3-point field goal
- Hypothetical game score: UCSD 75, UCLA 64
- Assuming no 3-pointers, in how many ways could UCSD have accumulated 75 points?

Notation:
- \( S(n) \equiv \# \text{ ways to score } n \text{ points} \)

Small Cases:
- \( S(0) = 1 \)
- \( S(1) = 1 \)
- \( S(2) = 2 \) (2 or 1-1)
- \( S(3) = 3 \) (2-1 or 1-2 or 1-1-1)

Is this familiar?

\[ \# \text{ways to reach } n \]
\[ = \# \text{ways to reach } n-2 \]
\[ \text{ (plus a field goal) } \]
\[ + \# \text{ways to reach } n-1 \]
\[ \text{ (plus a free throw) } \]
\[ \Rightarrow \text{Fibonacci numbers!} \]
Recurrence Relation

• **Problem:** What is S(75)?
  – **Notation:** write $F(n) = S(n-1)$
    
    $F(1) = F(2) = 1; \quad F(n) = F(n-1) + F(n-2)$

• **Fibonacci:** 1, 1, 2, 3, 5, 8, …
  So, $S(75)$ is the $76^{th}$ Fibonacci number

• **Solving the recurrence:** Next week

* Tuesday: start with fib1, fib2, fib3 and big-O asymptotic complexity
Choosing Between Solutions (Algorithms)

• Criteria:
  – Correctness
  – Time resources
  – Hardware resources
  – Simplicity, clarity (practical issues)

• Will need:
  – Size (“n”, number of bits, …), Complexity measures
  – Notion of “basic” (“unit-cost”) machine operation
The Basketball Question Again

• $\text{fib1}(n)$ if $n < 2$ then return $n$
  else return $\text{fib1}(n-1) + \text{fib1}(n-2)$
  – Analysis: $T(n) = 1$ if $n<2$; $T(n) = T(n-1) + T(n-2)$ otherwise
  $T(n) = F(n)$, i.e., around $(1.64)^n$

• $\text{fib2}(n)$ $f[1] = 1$; $f[2] = 2$;
  for $j = 3$ to $n$ do
    $f[j] = f[j-1] + f[j-2]$
  • Analysis: $T(n) = n$
    – Saving your work (“caching”) can be useful!

• Can we do “better”? (What is “better” anyway?)
Not Obvious, But Here Is A Shortcut…

• **fib3(n)**
  – Consider 2x2 matrix M: \( m_{11} = 0, m_{12} = 1, m_{21} = 1, m_{22} = 1 \)
  – Observe: 
    \[
    \begin{bmatrix}
    F(k) & F(k+1)
    \end{bmatrix}^T = M \times \begin{bmatrix}
    F(k-1) & F(k)
    \end{bmatrix}^T
    \]
    \[
    \begin{bmatrix}
    F(n+1) & F(n+2)
    \end{bmatrix}^T = M^n \times \begin{bmatrix}
    F(1) & F(2)
    \end{bmatrix}^T = M^n \times \begin{bmatrix}
    1 & 1
    \end{bmatrix}^T
    \]
  – How does this help?
  – **Hint:** \( 76_{10} = 1001100_2 \)

• \( M^{76} = M^{64} \times M^8 \times M^4 \)
• \( \rightarrow \) **fib3 uses “addition chains”**
Quantifying “Better”, “Worse” (Next Time)

• $2^n - 1$, $n \log_2 n + n$, $F_n$, … (how to assess, compare?)

• Resources used in computation often depend on a natural parameter, $n$, of the input
  – search/sort list       # items               $x > y$
  – matrix mult           largest dim       $x \times y ; x + y$
  – traverse tree          # nodes              follow ptr

• Asymptotic Notation     “as $n$ grows large”
  – $f \in O(g)$ if $\exists c_1, c_2 > 0$ s.t. $f(n) \leq c_1 g(n) + c_2 \quad \forall n > 0$
  – $f \in O(g)$ if $\exists c > 0, N$ s.t. $\forall n > N, f(n) \leq cg(n)$
    e.g., $200x^2 \in O(2x^{2.5})$
  – $f \in \Omega(g)$ if $g \in O(f)$
  – $f \in \Theta(g)$ if $g \in O(f)$ and $f \in O(g)$
Problems 13

- P13.1 An art gallery has the shape of a simple (non-self-intersecting) n-gon. What is the minimum number of watchmen needed to survey the building, no matter how complicated its shape? (The watchmen are placed at fixed posts. The building is surveyed when every point is visible to some watchman.)

- P13.2 A 6 x 6 rectangle is tiled by 2 x 1 dominoes. Prove that no matter how this is done, there is always a fault-line, i.e., a line that cuts across the rectangle without cutting any domino.