Lecture #24

CS170

Spring 2021

Lower Bounds

- Complexity of a problem P is a function $T(P)$ that measures its cost (time/memory/...) as a function $f(n)$ of its input size n
- An algorithm for P gives an upper bound on $f(n)$
  - Eg P = sorting, $T(P) = \#$ comparisons
  - Insertion Sort $\Rightarrow f(n) \leq O(n^2)$
  - Merge Sort $\Rightarrow f(n) \leq O(n \log n)$
- A lower bound for P is a proof that $f(n) = \Omega(g(n))$
  - Holds for any algorithm (in a class)
  - Eg: $f(n) = \Omega(n \log n)$ for #comparisons in sorting
  - $2^{\#\text{comparisons}} \geq n! \geq (n/2)^{(n/2)}$ (not bucket sort)

Lower Bounds

- Recall NP-complete problems: 3SAT, ILP, ...
  - widely believed lower bound: $\Omega(n^{\omega(1)})$
    i.e. bigger than any polynomial
  - Best known lower bounds: $\Omega(n)$
  - Proving lower bounds is hard
- CS172 preview
  - "Time Hierarchy Thm"
  - Given code implementing Boolean function C, that takes a binary string X of length $|X|=n$ as input, will C return true in $\le n^3$ steps?
  - Naive algorithm: run it for n^3 steps and see
  - Thm: Any correct algorithm takes $\Omega(n^{3-\varepsilon})$ steps for some tiny ε

Examples of Lower Bounds
for specific classes of algorithms

1) Circuit Complexity:
   - What size circuit (#wires or depth) is needed to solve a problem of input size n?
2) Cell Probe Model
   - How many reads/writes to memory are needed to solve a problem of size n?
3) Branching Program
   - How do time and memory needs trade off?
4) Communication Complexity
   - If Proc 1 owns X, and Proc 2 owns Y, how many bits do they need to exchange to compute $f(X,Y)$?

Circuit Complexity (1/13)

- Problem: Given $f:\{0,1\}^n \to \{0,1\}$, how big a circuit do you need to evaluate f?
  - Circuit = DAG of and, or, not gates
  - Size could be # wires, depth
- Ex $f:\{0,1\}^{10} \to \{0,1\}^{10}$ could multiply $x \cdot y$ where x = first 5 bits of input, y = last 5 bits
- What is known?
  - # circuits $= 2^{\Theta(w \log w)}$, w = # wires
  - # functions on n input bits $= 2^{2^n}$
  - $2^{c w \log w} \geq 2^{2^n} \Rightarrow C \cdot w \cdot \log w \geq 2^n$
  - Even if $w \sim 1.9^n$ most functions need more wires
  - Most $f:\{0,1\}^n \to \{0,1\}$ need exponentially many wires

Circuit Complexity (2/3)

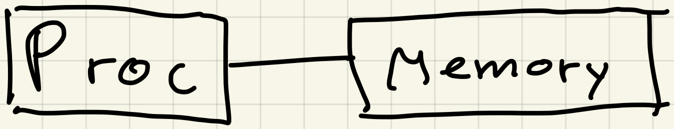
- Problem: Given $f:\{0,1\}^n \to \{0,1\}$, how big a circuit do you need to evaluate f?
  - Circuit = DAG of and, or, not gates
  - Size could be # wires, depth
- Most $f:\{0,1\}^n \to \{0,1\}$ need exponentially many wires
  - Do we know any?
  - Best result so far (2016): #wires $\geq (3+\frac{1}{86})\cdot n$
- What about depth?
  - f = Parity = XOR of all n input bits
  - Compute f using binary tree of depth $\log_2 n$
  - What if we restrict depth to a constant k? and allow unbounded fan in:

  - Thm: need #wires $= e^{\Omega(n)}$

Circuit Complexity (313)

- Problem: Given $f: \{0,1\}^n \to \{0,1\}$, how big a circuit do you need to evaluate f?
  - Circuit = DAG of and, or, not gates
  - Size could be # wires, depth
- Connection to NP-Completeness
  - Def: P/poly = set of all problems that can be solved with circuit of # wires $\leq$ poly(n)
  - P/poly is analogue of P for circuit complexity
  - Known: $P \subseteq P/poly$
  - Unknown: is $NP \subseteq P/poly$? If not, $P \neq NP$

Cell Probe Model

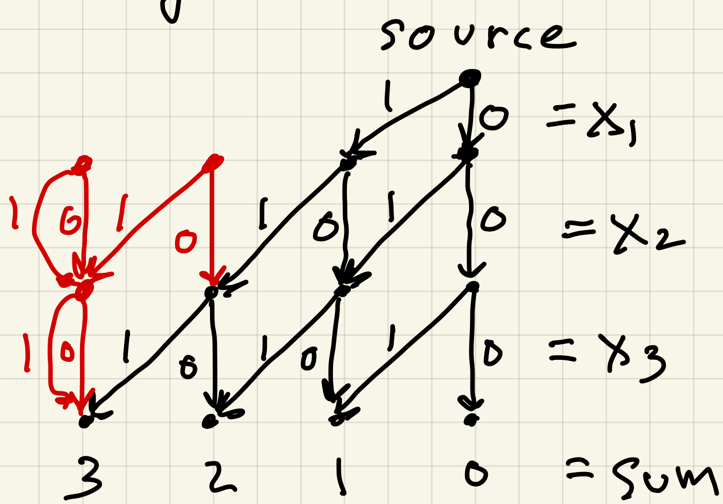
- Algorithm is allowed to perform the following ops:
  - 

  - Processor can read a word (w bits) from memory location i, or write a word
  - How many reads/writes to memory are needed to solve a problem of size n?
- Used to find lower bounds on cost of using data structures

Branching programs

- DAG to compute $f:\{0,1\}^n \to Y$
  - one source node
  - one sink node per element of Y
  - each non-sink nodes has 2 outgoing edges labelled $x_i = 0$ or $x_i = 1$, where x = input

- Adding 3 bits: $Y = \{0, 1, 2, 3\}$



Width $= 4 = 2^{\#\text{bits needed}}$

$\#$Layers $= 3 = \#$steps to compute answer $\approx$ runtime

- How do Width(#bits) and #Layers(runtime) tradeoff?

Communication Complexity (1/8)

- Alice and Bob both want $f:(X,Y) \to \{0,1\}$
  - Alice only knows X, Bob only knows Y
  - They exchange messages, last one to receive a message announces $f(X,Y)$
  - Goal: minimize #bits Alice and Bob exchange
- Different kinds of algorithms allowed:
  - Deterministic
  - Public coin ramdomness: Alice and Bob have same random bits
  - Private coin randomness: Alice and Bob have their own random bits
  - One way communication (Alice sends one message to Bob) vs. 2-way

Communication Complexity (2/8)

- Alice knows X, Bob knows Y, want $f(X,Y) \in \{0,1\}$ while minimizing # bits exchanged
- Def: $D(f)$ = minimum # bits with deterministic alg
- Def: $R_{pub}(f)$ = minimum # bits with public coin randomness
- Def: $R_{priv}(f)$ = minimum # bits with private coin randomness
- Thm: $D(f) \geq R_{priv}(f) \geq R_{pub}(f)$

  Proof: 1st ineq: deterministic special case: ignore random bits

  2nd ineq: Alice uses 1st half of random bit, Bob 2nd half
- Def: $EQ(X,Y) = 1$ if $X=Y$ else 0
- Thm: $D(EQ) = \Theta(n)$, $R_{priv}(EQ) = \Theta(\log n)$, $R_{pub}(EQ) = \Theta(1)$
- Simplify: consider one-way communication cost, $D^{\rightarrow}(f)$, Alice sends one message to Bob

Communication Complexity (3/8)

- Alice knows X, Bob knows Y, want $f(X,Y) \in \{0,1\}$
- Def: $\vec{D}(f)$ = minimum #bits with deterministic alg where Alice sends one message to Bob
- Claim: $\vec{D}(EQ) \geq n = |X|$
  proof: If Alice sends $g(X) \in \{0,1\}^m$ to Bob with $m<n$, $\exists\ X_1 \neq X_2$ but $g(X_1) = g(X_2)$ so Bob can't tell the difference between X_1 and X_2
- Def: DE = counting # distinct elements
- Claim: Any exact deterministic alg A for DE requires $\Omega(n)$ bits of memory (FM was random)

proof: Show that if A solves DE with s bits, we can use A to solve EQ with $s + \log n$ bits, so
$$s + \log n \geq \vec{D}(EQ) \geq n \Rightarrow s \geq n - \log n = \Omega(n)$$

Communication Complexity (4/8)

- Alice knows X, Bob knows Y, want $f(X,Y) \in \{0,1\}$
- Def: $D^{\rightarrow}(f)$ = minimum #bits with deterministic alg where Alice sends one message to Bob
- Claim: $D^{\rightarrow}(EQ) \geq n = |X|$
- Def: DE = counting # distinct elements
- Claim: Any exact deterministic alg A for DE requires $\Omega(n)$ bits of memory (FM was random)

proof: Show that if A solves DE with s bits, we can use A to solve EQ with $s + \log n$ bits, so

$$s + \log n \geq D^{\rightarrow}(EQ) \geq n \Rightarrow s \geq n - \log n = \Omega(n)$$

How to use A to solve DE: For each $X_i = 1$, Alice feeds i into A, sends (mem(A), #1s in X) = $s + \log n$ bits to Bob. Bobs checks if #1s in X = #1s in Y. If yes, Bob ask A for #DE, then feeds i for each $Y_i = 1$ into A and asks if #DE increases. If not, $X = Y$, else $X \neq Y$

Communication Complexity (5/8)

- Claim: Any exact deterministic alg A for DE requires $\Omega(n)$ bits of memory
- Claim: Any approximate deterministic alg for DE ($|\tilde{t}-t| \le .01t$) also requires $\Omega(n)$ bits of memory

Proof: consider EQ where $X, Y \in B \subseteq \{0,1\}^n$

Same argument as before $\Rightarrow D^{\rightarrow}(EQ^B) \ge \log_2 |B|$

Claim: (no proof!) $\exists$ B such that

$|B| \ge 2^{cn}$, all $X \in B$ have same #1s $= r$,

and if $X, Y \in B$, $X \ne Y$, then #1s X and Y share $\le \frac{r}{10}$

(Think of $X, Y \subseteq \{1, \dots, n\}$, so $|X| = |Y| = r$, $|X \cap Y| \le \frac{r}{10}$)

Show that if A_{ap} solves DE with s bits, 1% error,

we can use A_{ap} to solve EQ^B with s bits

$\Rightarrow s \ge D^{\rightarrow}(EQ^B) \ge \log_2 |B| \ge cn = \Omega(n)$

Communication Complexity (6/8)

- Claim: Any approximate deterministic alg for DE ($|\tilde{t}-t| \le .01t$) also requires $\Omega(n)$ bits of memory

Proof: EQ where $X, Y \in B \subseteq \{0,1\}^n$, $D^{\rightarrow}(EQ^B) \ge \log_2 |B|$

Claim: (no proof!) $\exists$ B such that

$|B| \ge 2^{cn}$, all $X \in B$ have same #1s $= r$,

and if $X, Y \in B$, $X \ne Y$, then #1s X and Y share $\le \frac{r}{10}$

Show that if A_{ap} solves DE with s bits, 1% error,

we can use A_{ap} to solve EQ^B with s bits

$\Rightarrow s \ge D^{\rightarrow}(EQ^B) \ge \log_2 |B| \ge cn = \Omega(n)$

How to use A_{ap} to solve EQ^B (similar to using A for EQ):

For each $X_i = 1$, Alice feeds i into A_{ap}, sends mem(A_{ap}) to Bob.

Bob knows #DE $= r$ (property of B). For each $Y_i = 1$, Bob feeds i into A_{ap}, asks for #DE. 2 cases:

$X = Y \Rightarrow A_{ap}$ reports #DE $\in [.99r, 1.01r]$

$X \ne Y \Rightarrow A_{ap}$ reports #DE $\in [.99 \cdot 1.9 \cdot r, 1.01 \cdot 2 \cdot r]$

different!

Communication Complexity (7/8)

- Summary of counting distinct elements (DE)

  No exact, deterministic alg with $o(n)$ memory

  No approx, deterministic alg with $o(n)$ memory

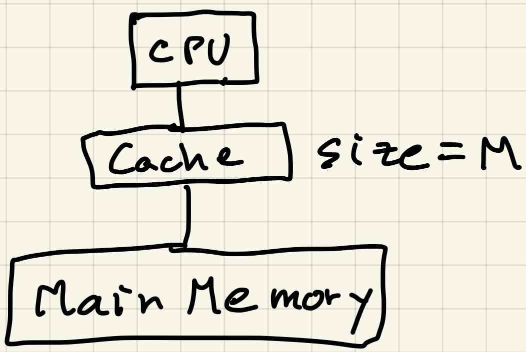
  No exact, randomized alg with $o(n)$ memory

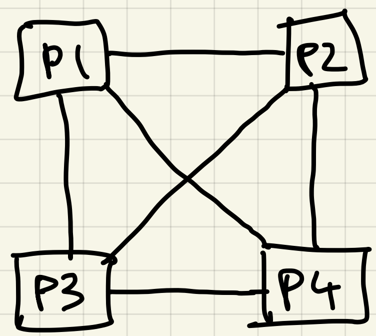
  $\Rightarrow$ need approx, randomized alg to use $o(n)$ memory (FM)

- How to show $\mathcal{B}$ exists? choose randomly, show it has right property with probability > 0

Communication Complexity (8/8)

- Goal: minimize communication between main memory and cache, or between processors connected over a network





- Thm (Hong, Kung, 81) Any execution of $\Theta(n^3)$ matrix multiply moves $\Omega(n^3/\sqrt{M})$ words between Cache and main memory
- Attained by "looptiling", widely implemented
- Extends to rest of linear algebra, any code that looks like nested loops accessing arrays