(pp. 117-124) **Non Context Free Languages** (Sec. 2.3)

- (p. 125) **Pumping Lemma for CFLs**
  - IF A is a CFL, then for some # p (**pumping length**)
    - If s is *any* string in A, |s| ≥ p
    - Then s = u**vxy**z (for some 5 substrings u,v,x,y,z) where
      - For all i≥0, uv^ixy^iz is in A
      - And |**vy**| > 0
        - the length of the 2 pumped parts v and y is not 0
        - but just one of v or y could be ε
      - and |**vxy**| ≤ p
        - The middle string x is at most the pumping length
  - Using this lemma: if we can find even one string from L where
    - there is no possible partitioning into 5 pieces
      - i.e. we look at all possible partitionings
    - where all conditions hold (esp. the first)
    - then L is not CFL

- (p. 124) Notional proof
  - If L is CFL then we can draw a parse tree like (p. 126) Fig. 2.35 (a) to generate each string in L
  - Pick a string “long enough” that we have to reuse one of the non-terminals, say R
  - The derivation between the 1st point where R is in the tree and its reuse could then be substituted over and over (Fig. 2.35 b) for the second use, or not at all (Fig. 2.35c)
- Example: develop language for, and then draw parse tree for S->aSb S->#

- Estimating pumping length p
  - Let G be CFG for A
  - Let **b** = max # of variables on any rule RHS
  - Thus, in any parse tree, no interior node (variable) can have more than b children.
    - So at most b leaves are one step from start variable
    - At most b^2 children 2 steps from start
    - At most b^3 children 3 steps from start
    - ….
    - Or, *<u>at most</u>* b^h leaves from start in tree of h levels
    - **OR: if height of parse tree ≤ h then string length ≤ b^h**
    - **OR: If $|s| \geq b^h + 1$, then parse tree at least h+1 high**
  - Now assume $p = b^{|V|+1}$
    - If $|s| \geq p$ then parse tree must be at least $|V|+1$ high
    - So some R must have been used more than once
      - For convenience select R as 1st one that repeats among lowest $|V+1|$ variables on longest path
  - Upper occurrence of R generates **vxy**
  - Lower occurrence of R generates **x**
  - Replacing the lower by the upper "pumps up"
  - Replacing upper by lower "pumps down"
  - All must be in A because generated by G

- (p. 128) Example B = $\{a^n b^n c^n \mid n \geq 0\}$ not CFL
  - Assume B CFL so there is some p
  - Select s = $a^p b^p c^p$ (we need only 1 string for contradiction)
    - Clearly in B with length >p
  - Pumping lemma says no matter how we divide s in u**vxyz**, one condition fails
    - Either v or y must be non empty
  - Two cases
    - Only one kind of terminal in v and y
      - Then x must be same terminal
      - And then vxy must be in one of 3 parts a^n, b^n, or c^n
        - And thus all characters in vxy are same
      - And then pumping v and y (one is non empty) destroys balance
    - When either v or y contain more than one type of terminal, then uv^2xy^2z might contain right #s but not all grouped together.

- (p. 128) Example C = $\{a^i b^j c^k \mid 0 \le i \le j \le k\}$ not CFL
  - Consider s = $a^p b^p c^p$
- (p. 129) Example D = {ww| w in {0,1}*} not CFL
  - Consider s = $0^p 1^p 0^p 1^p$
    - Must straddle midpoint
    - Then it distorts trailing 1s on left from trailing 1s on right
- See also problems 2.30-2.33, 2.45,