Admin:

Reading: CLRS 15.1–15.4

Outline: Dynamic Programming (2/4)

- ☐ Review DP concepts
- ☐ Longest path in a DAG
- ☐ Longest common subsequence
- ☐ Picking up pennies

Review Dyn. Prog. concepts

- typically, an optimization problem that can be solved recursively.
- recursive calls generate similar subproblems
- subproblem dependence graph: (DAG: no cycles!)

  V = set of subproblems

  $E = \{ x \to y$: solving x involves calling for y's solution$\}$

- Two approaches:

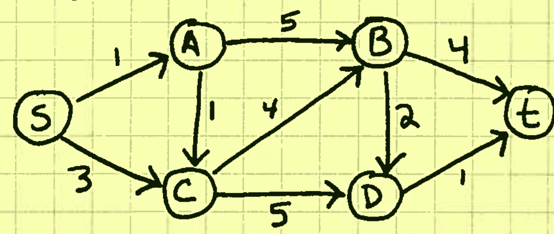
  ① "top-down" (recursive) implementation, but "memoizing" solutions to subproblems as they are solved] ≡ DFS

  ② "bottom-up": sort V in reverse topological order, solve subproblems in that order, save solutions as subproblems are solved.

- Solution time is typically $\Theta(V+E)$, assuming that solving a subproblem is linear in number of recursive calls it makes (i.e., in # of outgoing edges it has).

Longest Path in a DAG

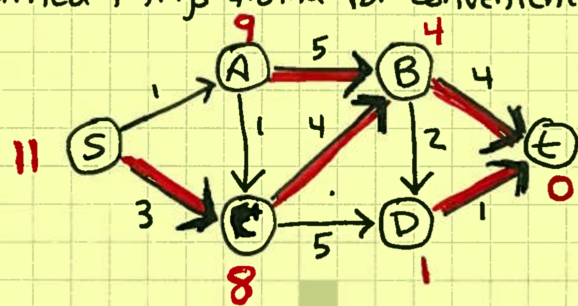
- Find longest $s \to t$ path in given DAG $G = (V, E)$



- subproblem for each vertex u:

  what is length of longest $u \to t$ path, and
  what is first vertex (after u) on this path?

("single destination" rather than "single source" — we've turned things around for convenience, no big deal.)



$$d[u] = \max_{(u,v) \in E} \left(w(u,v) + d[v] \right) \qquad (\text{but } d[t] = 0)$$

- do DFS on G, starting at s.
  save ("memoize") $d[u]$ when u "finished". (illustrate)
  time is $\Theta(V+E)$

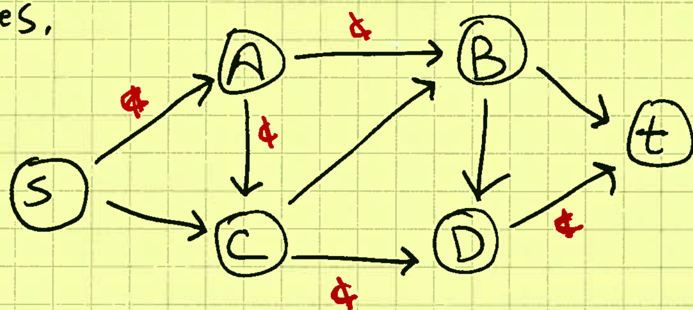
- can find "critical path" in project ("PERT chart") similarly

$$d[u] = w(u) + \max_{(u,v) \in E} \left(d[v] \right) \qquad (\text{but } d[t] = w(t))$$

where $w(u)$ = time to complete task u on its own.

"Picking up pennies"

Given a DAG $G = (V, E)$, where some edges have pennies on them & some don't, find path from given start vertex with most pennies.



(Ans = sACDt)

This is just longest path problem, where length of edge = # of pennies on it!

6.006
Rivest
L20.5
11/13/08

Longest Common Subsequence (LCS)

- apps: "edit distance", "diff", spell-checking, DNA comparison, document distance (new def), plagiarism detection, etc.

- Given two strings/sequences x & y, find a longest common subsequence (elements in same order but not nec. contiguous)

- Example: $x = $ C A T G $\quad$ or $\quad x = $ C A T G
  $y = $ A C G $\qquad\qquad\qquad\quad y = $ A C G
  $z = LCS(x,y) = AG \qquad\qquad z = LCS(x,y) = CG$
  (may be more than one LCS)

- Naive brute force: try all $2^{|x|}$ subsequences of x
  see if it occurs in y
  $$\text{time} = \Theta\left(2^{|x|} \cdot |y|\right)$$

- Consider "chopping game" - try to maximize # points
  Given two strings, do a sequence of moves until nothing is left.

  Move = drop 1^{st} char of x (0 pts)

  CATG

  or drop 1^{st} char of y (0 pts)

  ACG

  1 1

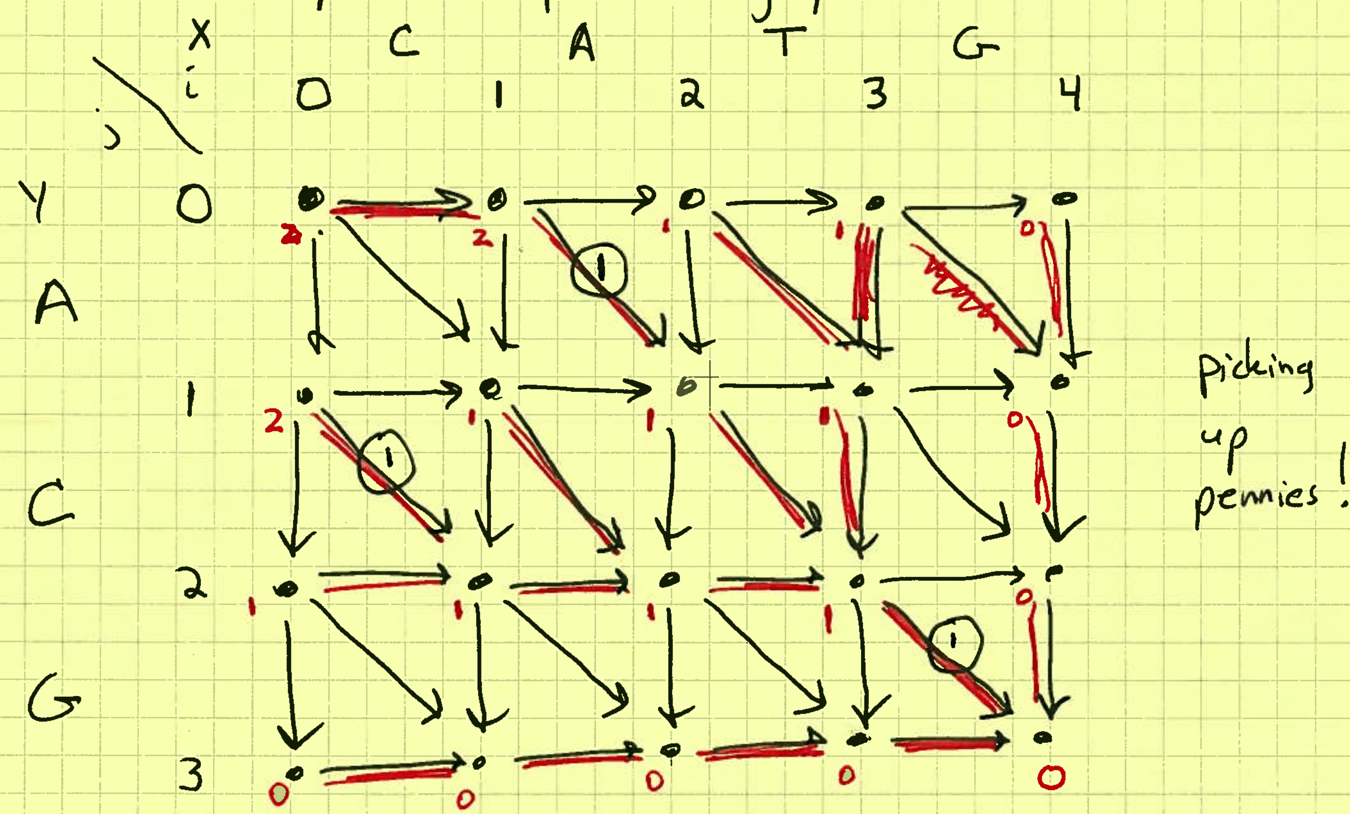
  or drop 1^{st} chars of x & y (1 pt if same char dropped else 0 pts)

- Claim: max score achivable = length of LCS
  (Just take letters corresponding to points achieved.)

6.006
Rivest
L20.6
11/13/08

- State of game = subproblem
  = pair of suffixes $(x[i:], y[j:])$
  after having dropped i from x & j from y

- Draw subproblem dependence graph



$\longrightarrow$ ≡ drop x

$\downarrow$ ≡ drop y

$\searrow$ ≡ drop both

- all edges have 0 weight except for $\searrow$ with $x[i] = y[j]$
- want <u>longest path</u> (!)

$$\text{Time} = \Theta(V+E) = \Theta(|x|\cdot|y|)$$

$$\text{Space} = \Theta(|x|\cdot|y|)$$

$(\Rightarrow \Theta(\min(|x|,|y|))$
just to find length
do by rows, or columns$)$