6.006
Rivest
L22.1
11/20/08

Admin:

Readings: CLRS 15.1-15.4

Outline: Dynamic Programming (3/4)

- ☐ matrix chain multiplication (finish)
- ☐ printing text neatly
- ☐ vertex cover
- ☐ wrap-up

6.006
Rivest
L22.2
11/20/08

Matrix Chain Multiplication

Given: $A_1\ A_2\ A_3\ \cdots\ A_n$ matrices

rows: $r_1\ r_2\ r_3\ \cdots\ r_n$

cols: $c_1\ c_2\ c_3\ \cdots\ c_n$ (where $c_i = r_{i+1}$ for $1 \le i < n$)

Find: way to evaluate their product most efficiently
($\equiv$ optimal parenthesization of this expression)

Fact: Time to multiply an $(r \times c)$ by $(r' \times c')$ matrix is $\Theta(rcc')$
(where $c = r'$), using naive method.

$A_1 (A_2 \cdot A_3)$ $\quad$ A_1 / $A_2\ A_3$ $\quad$ cost $= r_2 c_2 c_3 + r_1 c_1 c_3$

Subproblems: compute product $A_i \cdot A_{i+1} \cdots A_j$ optimally

Let m_{ij} = cost of this computation

$$m_{ii} = 0 \qquad \text{(no product to do; only one matrix)}$$

$$j > i \Rightarrow m_{ij} = \min_{i \le k < j} \left(m_{ik} + m_{k+1,j} + r_i c_k c_j \right)$$

m_{ik}: for $A_1 \cdots A_k$ $\quad$ $m_{k+1,j}$: for $A_{k+1} \cdots A_j$ $\quad$ $r_i c_k c_j$: to mpy them together

$$\#\text{Subproblems} = n + \binom{n}{2} = \Theta(n^2) = \#\text{vertices in subproblem dep. graph}$$

$$\#\text{edges} = \sum_{1 \le i \le j \le n} 2 \cdot (j - i)$$

$$= \Theta(n^3)$$

6.006
Rivest
L22.3
11/20/08

So running time to find optimal parenthesization of $A_1 \cdot A_2 \cdots A_n$ is $\Theta(V+E) = \Theta(n^3)$.

(Don't confuse cost ↑ of computing optimal with cost of then actually computing product itself!)

6.006
Rivest
L22.4
11/20/08

Printing text neatly

Given: a paragraph of n words of length ℓ_i $1 \le i \le n$
an integer W (line width) $[W \ge \max_i \ell_i]$

Find: a way to print words on lines so that

- printed in order
- if $\ell_i \ldots \ell_j$ on one line then this takes $\ell_{ij} \triangleq \underbrace{\ell_i + \ell_{i+1} + \cdots + \ell_j}_{\text{for words}} + \underbrace{(j-i)}_{\text{for separating blanks}}$

  which must be $\le W$,
- followed by $t_{ij} = W - \ell_{ij}$ trailing blanks
- so as to minimize sum of cubes of # trailing blanks on each line (except last line ignored in this measure).

Example: text = "A CAT IN THE HAT" $W = 6$

1.	A␣CAT␣	A␣␣␣␣␣␣
2.	IN␣THE	CAT␣IN
3	HAT␣␣␣	THE␣␣␣
4		HAT␣␣␣
Cost	$1^3 = 1$	$5^3 + 3^3 = 152$

"Greedy" approach packs each line as fully as possible, before moving on to next line.

This minimizes # lines used, but not our measure.

Example: "CAT IN A BASKET" $W = 6$

CAT␣IN		CAT␣␣␣
A␣␣␣␣␣	vs	IN␣A␣␣
BASKET		BASKET

(greedy)
cost = $5^3 = \underline{\underline{125}}$

(optimal)
cost = $3^3 + 2^3 = \underline{\underline{35}}$

But DP can solve this problem optimally.

Subproblem = $(\ell_i, \ell_{i+1}, \ldots, \ell_n)$, find min cost ($\Theta(n)$ subproblems)

$$\text{cost}(\ell_i, \ldots, \ell_n) = 0 \quad \text{if } \ell_{in} \le W \quad [\text{fits on one line}]$$

$$\text{else} = \min_{\substack{i \le k < n \\ \text{s.t.} \\ \ell_{ik} \le W}} \left[(W - \ell_{ik})^3 + \text{cost}(\ell_{k+1}, \ldots, \ell_n) \right]$$

$(W - \ell_{ik})^3$: cost on 1st line

$\text{cost}(\ell_{k+1}, \ldots, \ell_n)$: cost of rest of paragraph

edges = $O(n^2)$

total time = $O(V+E) = O(n^2)$ ← probably much less...

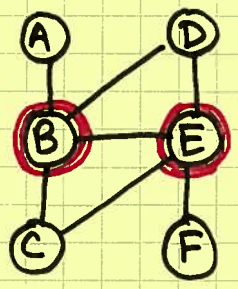
6.006
Rivest
L22.6
11/20/08

Vertex Cover

Given an undirected graph $G=(V,E)$

find a minimum set of vertices (a "cover") such that each edge is covered on ≥ 1 end.

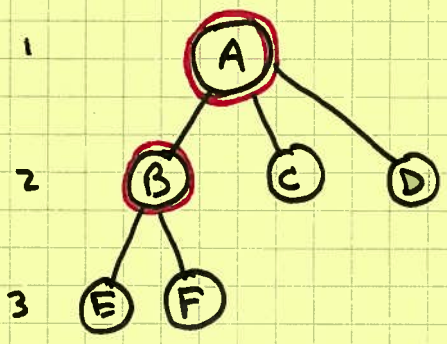
Example:



min cover has size 2

$\{B, E\}$

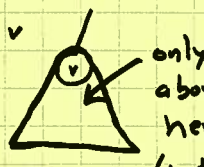
NP-complete for general graphs, but polynomial for trees, with DP.



Note: picking vertices on odd levels $\{A, E, F\}$
or on even levels $\{B, C, D\}$
gives cover, but doesn't give
minimum cover $\{A, B\}$

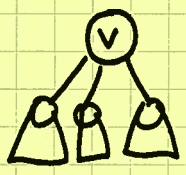
For DP, subproblems are subtrees rooted at a vertex v

$\Rightarrow |V|$ subproblems

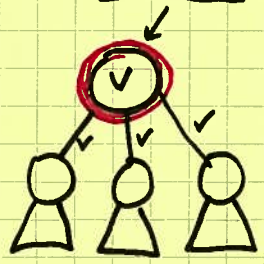


Subproblem dependence graph is same as G, except,
as we'll see, we add edges from v to its grandchildren as well
(Exercise: show # added edges is $O(V)$)

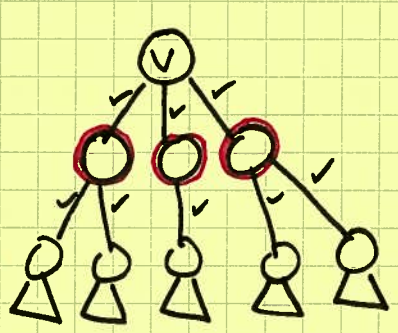
6.006
Rivest
L22.7
11/20/08

Consider subtree 

either v is in cover, or it is not



recurse on children



recurse on grandchildren

Let $f(v)$ = min size of cover for subtree rooted at v.

$$= \min\Big(1 + \sum_{c} f(c) \quad \text{for } c \text{ a child of } v, \qquad v \text{ taken}$$

$$\#\text{children}(v) + \sum_{g} f(g) \quad \text{for } g \text{ a grandchild of } v \Big) \qquad v \text{ not taken}$$

Time $= O(V+E) = O(V)$ (since G is a tree, & s.d.g. adds $O(V)$ edges)

$f(\text{root})$ gives answer, can reconstruct as usual by storing flag as to which way min was found

6.006
Rivest
L22.8
11/20/08

Dynamic Programming

- powerful technique for solving many kinds of optimization problems
- saves solution to subproblems, for reuse later
- yields $O(V+E)$ running time typically, where $G=(V,E)$ is ~~size~~ subproblem dependence graph.