

optimal substructure

- make a choice
- recursively have subproblem(s)
- problems must exhibit optimal substructure

Complexity

- how many subproblems in an optimal sol'n
- # of choices for a subproblem
- total # of distinct subproblems

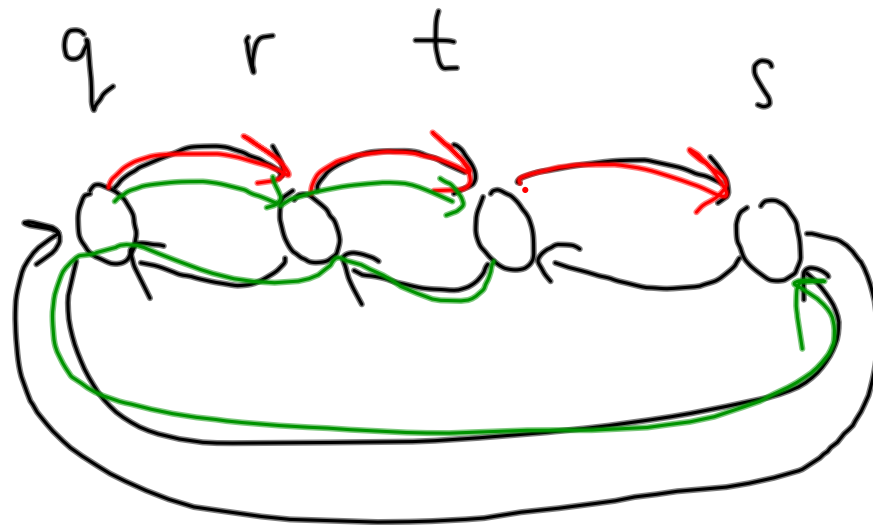
<u>C</u>	<u>P</u>	<u>M</u>
1	1	2
k (#denom)	3	n
n	n^2	n^2
nk	n^2	n^3

Shortest path
opt. substt.



longest simple path

Given a directed graph, find
the longest path from x to y
that does not visit any vertex
more than once



q to s longest simple path

$LSP(q, s)$ consist of $LSP(q, t)$ and $LSP(t, s)$

(() (...))
↑ ↑
Indep. of each other

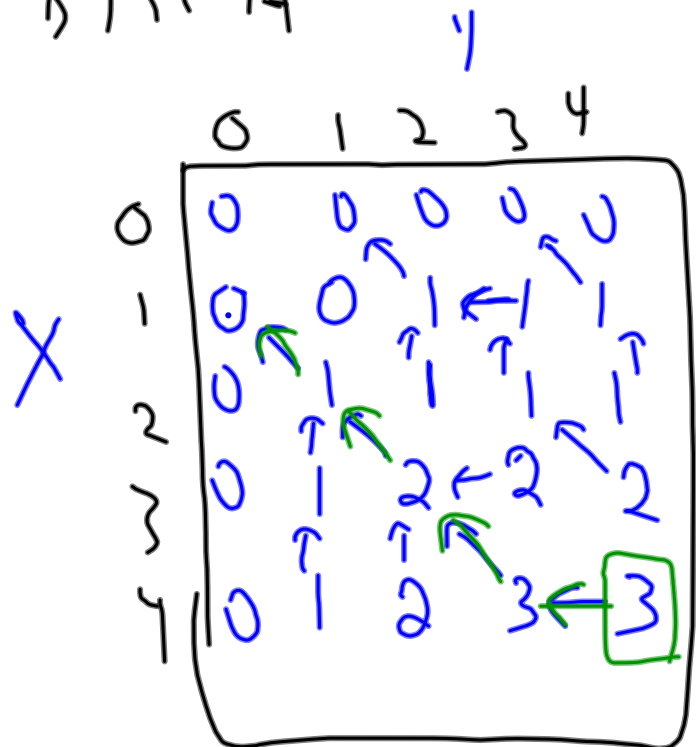
-
- Recursive sol'n
 - see limited # of subproblems
 - compute subproblems "bottom-up"

Can you automated figuring out
where the bottom is?
YES, but the tradeoff is unclear

Memoization - 'automatically'
keep track of which subproblems
you have solved.

X = AB A C

Y = B A C A



BAC

$O(nm)$ time