NANYANG TECHNOLOGICAL UNIVERSITY

SC2001/ CX2101: Algorithm Design and Analysis

Week 11

Huang Shell Ying

Another example of the longest path problem not satisfying the principle of optimality

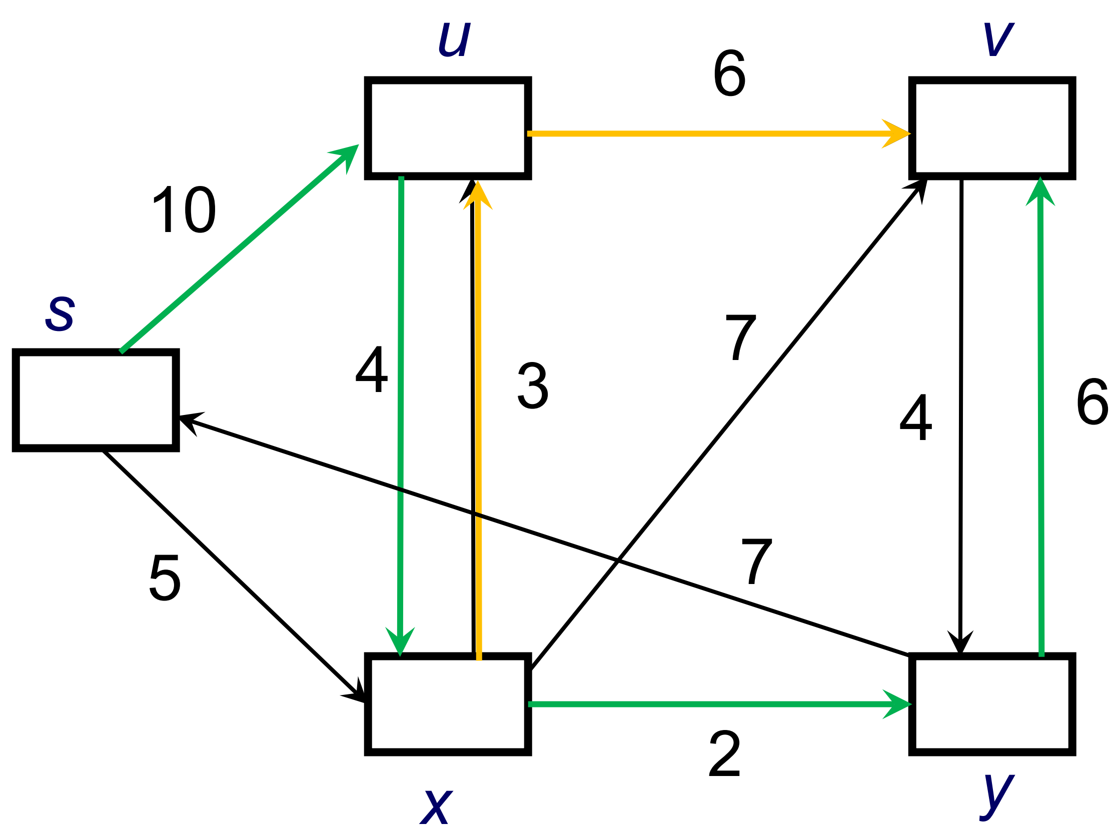
The longest path from s to v is

s -> u -> x -> y -> v

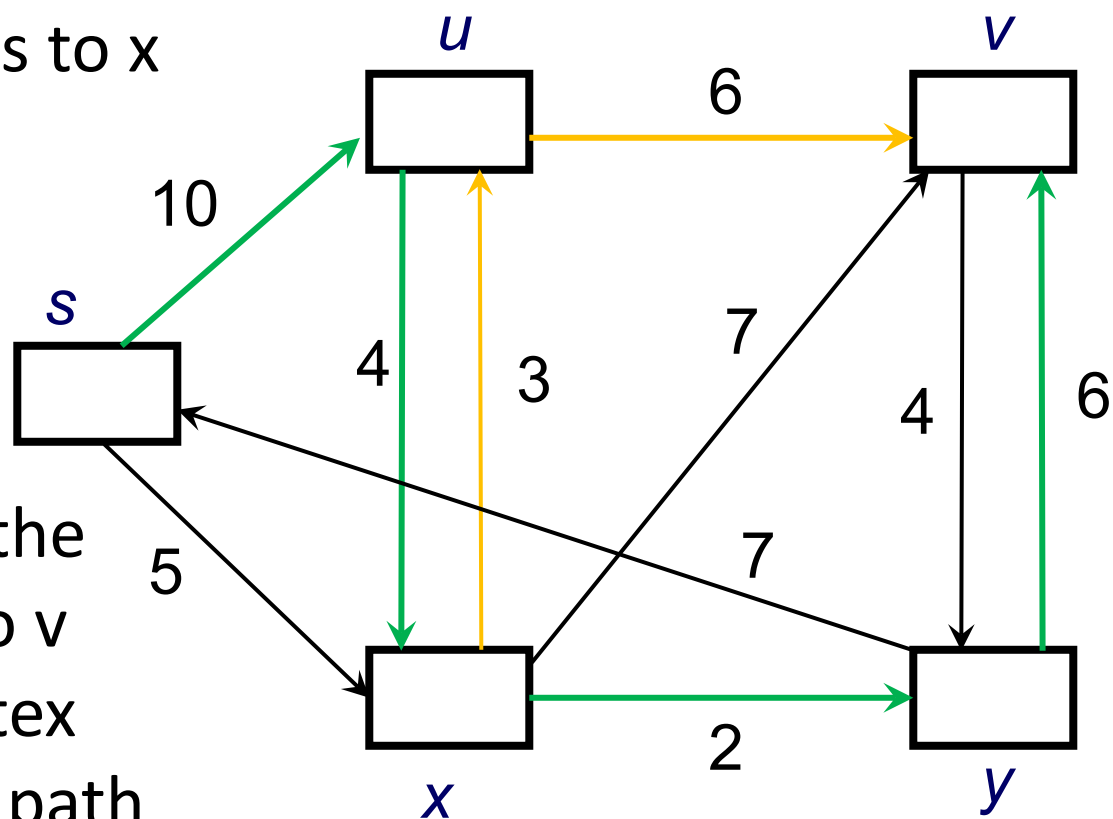
The longest path from x to v?

Why?

Note, the longest path means the longest simple path (path with no cycle)



The solution to the problem of the longest path from s to v includes the solutions of 2 subproblems: from s to x and from x to v.



The longest path for the subproblem from x to v passes through a vertex which appears in the path from s to x

Summary: Greedy vs DP

A greedy strategy for coin change problem

Example: Making Change

Problem: A country has coins with denominations

$$1 = d_1 < d_2 < \cdots < d_k.$$

You want to make change for n cents, using the smallest number of coins.

Example: U.S. coins

$$d_1 = 1 \;\; d_2 = 5 \;\; d_3 = 10 \;\; d_4 = 25$$

Change for 37 cents – 1 quarter, 1 dime, 2 pennies.

What is the algorithm?

A greedy strategy for coin change problem

```
c = 0;
While (n> 0)  {
    c++;
    n = n – value of the highest valued coin that is <= n;
}
Return c;                         // O(n)
```

An even faster version of the greedy approach is

1. $a_1 = \lfloor n/d_k \rfloor$, $n = n - a_1 * d_k$
2. $a_2 = \lfloor n/d_{k-1} \rfloor$, $n = n - a_2 * d_{k-1}$
3. …

// O(k), or O(1) if k is a known constant

A greedy strategy for coin change problem

- Works for many cases, e.g. to change for 37 cents,

$$d_1 = 1 \quad d_2 = 5 \quad d_3 = 10 \quad d_4 = 25$$

The answer is optimal: one 25¢, one 10¢, two 1¢

- but fails for cases like d ={1, 10, 25} and we want to change 30 cents

A Greedy Method for chain matrix multiplication

1. use an array to record the dimensions.

E. g A_1 x A_2 x A_3 x A_4

	0	1	2	3	4
d	30	1	40	10	25

2. choose the multiplication of two matrices whose cost is the minimum at each step:

First, min(30x1x40, 1x40x10, 40x10x25) $\Rightarrow$ 1x40x10 is the minimum. So A_2x A_3 first:

0	1	2	3	4
30	1	~~40~~	10	25

A Greedy Method for chain matrix multiplication

0	1	2	3	4
30	1	~~40~~	10	25

Second, min(30x1x10, 1x10x25) $\Rightarrow$ 1x10x25 is the minimum. So (A_2x A_3)x A_4

0	1	2	3	4
30	1	~~40~~	~~10~~	25

Last: 30x1x25: A_1((A_2x A_3)x A_4)

Total : 1x40x10 + 1x10x25 + 30x1x25 = 1400

Works in most cases except some sequences of 3 matrices (i.e. 2 matrix multiplications) e.g.
A_1x A_2x A_3 : 10x1x10x15 $\Rightarrow$ 10x1x10 + 10x10x15

Another Greedy Method for chain matrix multiplication

1. use an array to record the dimensions.

E.g A_1 x A_2 x A_3 x A_4

d

0	1	2	3	4
30	1	40	10	25

2. choose the multiplication of two matrices whose resulting matrix has minimum dimension at each step:

First, min(30x40, 1x10, 40x25) $\Rightarrow$ 1x10 is the minimum. So A_2x A_3 first:

0	1	2	3	4
30	1	~~40~~	10	25

Another Greedy Method for chain matrix multiplication

0	1	2	3	4
30	1	~~40~~	10	25

Second, min(30x10, 1x25) $\Rightarrow$ 1x25 is the minimum. So $(A_2 x A_3) x A_4$

0	1	2	3	4
30	1	~~40~~	~~10~~	25

Last: 30x25: $A_1((A_2 x A_3) x A_4)$

Total : 1x40x10 + 1x10x25 + 30x1x25 = 1400

Works in most cases but not for this (question in the tutorial T4Q5)

d	20	2	15	40	4

$A_1 x A_2 x A_3 x A_4 : A_1 x (A_2 x (A_3 x A_4)) \Rightarrow$ 2680 (1680 by DP)

Summary: Greedy vs DP

- DP is an **optimization** technique
- Both build solutions through a sequence of individual steps
- Both make a selection out of a collection of choices at each step
- At each step, the greedy method computes its **locally optimal** choice one after another and never revises any choices made – optimal solution not guaranteed, but many problems do get the optimal solution
- At each step, DP computes its solution by trying all choices before it arrives at the optimal choice

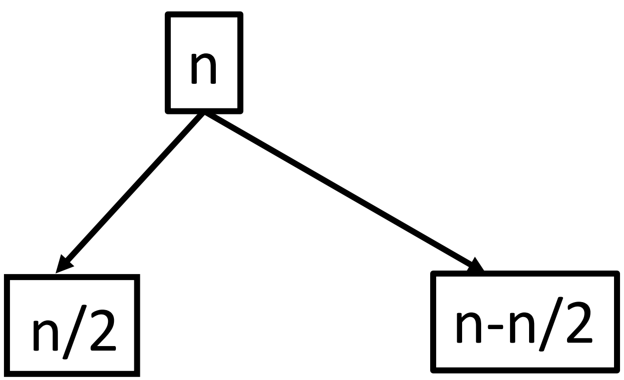
- Greedy heuristic algorithms are often used to solve problems because of its simplicity.
- Typically, DP algorithms are more expensive than greedy algorithms
- So DP is used only when no greedy strategy can be found to deliver the optimal solution

Summary: Divide-and-conquer vs DP

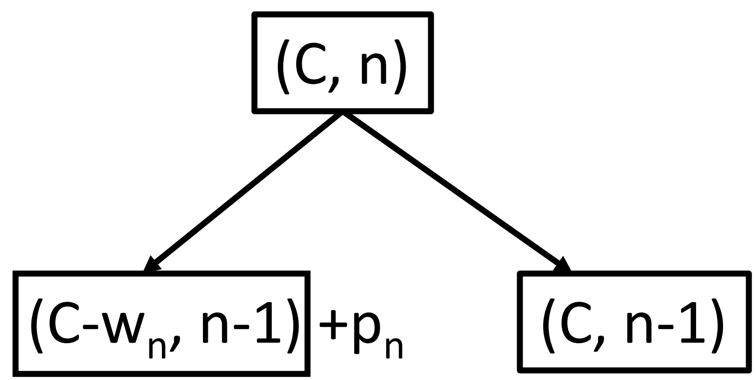
Summary: Divide-and-conquer vs DP

- Both techniques divide their problems into subproblems, find subsolutions to the subproblems, and synthesize larger solutions from smaller ones.

mergeSort



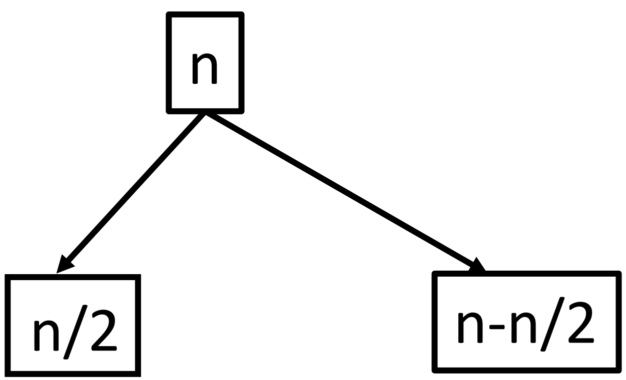
DP for Knapsack



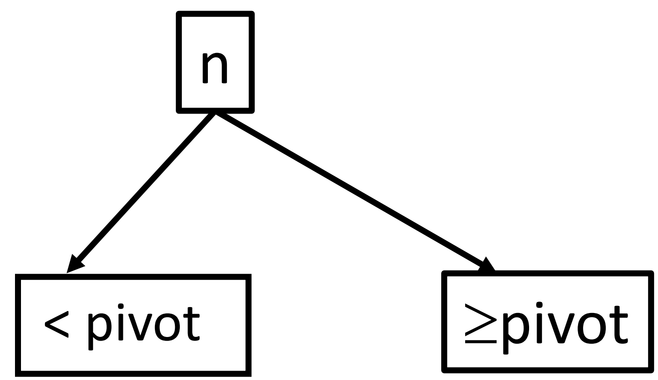
Summary: Divide-and-conquer vs DP

- Divide and Conquer divides a problem at prespecified deterministic points (e.g., always in the middle), combine the subsolutions to obtain the solution to a larger problem

mergeSort



quickSort

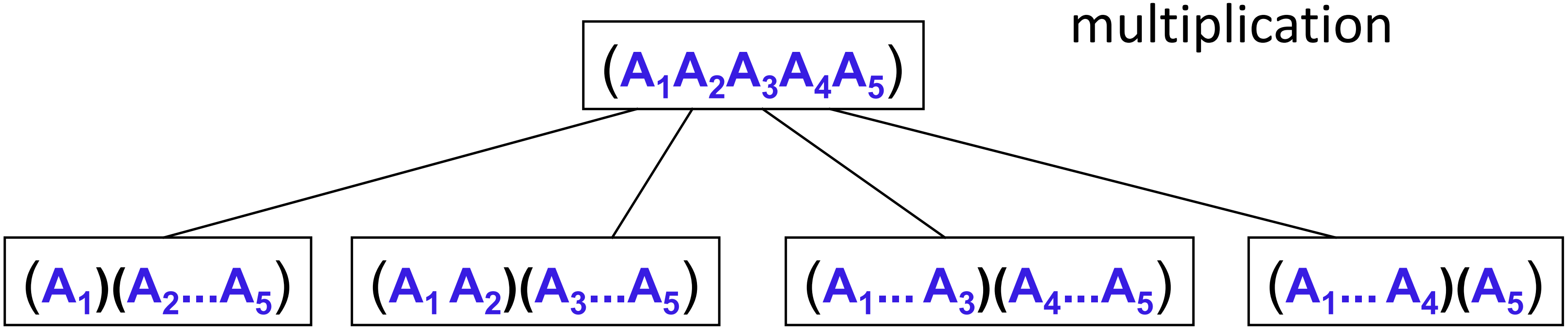


Divide at deterministic point

Summary: Divide-and-conquer vs DP

- For optimization problems, DP divides a problem at every possible split points rather than at a pre-specified points. After trying all split points, it determines which split point is optimal. The subsolution at the split point is part of the solution to a larger problem

DP for chain matrix multiplication



Divide at all possible split point