Dynamic Programming Concluded

April 9–10, 2012

CS 60: Principles of Computer Science

Assignment 10 (Dynamic Programming) due Monday, April 16.

Recall: Fibonacci Numbers

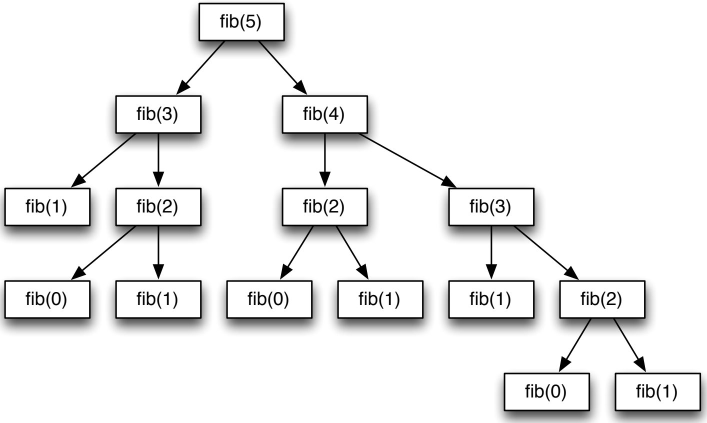
The Fibonacci numbers are easy to compute!

$$\text{fib}(n) = \begin{cases} n & \text{if } n = 0 \text{ or } n = 1 \\ \text{fib}(n-2) + \text{fib}(n-1) & \text{if } n \geqslant 2 \end{cases}$$

The corresponding recursive code is exponentially slow for large n.

The Problem

$$\text{fib}(n) = \begin{cases} n & \text{if } n = 0 \text{ or } n = 1 \\ \text{fib}(n-2) + \text{fib}(n-1) & \text{if } n \geq 2 \end{cases}$$

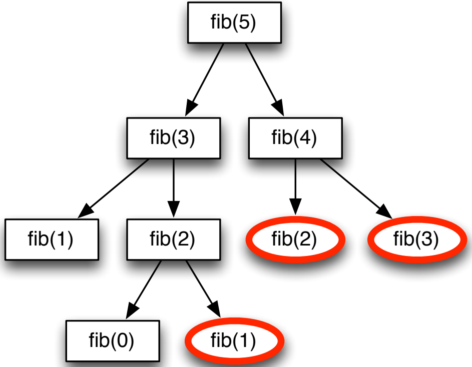


Idea 1: "Memoizing"

Remember all the inputs and output so far

Return precomputed answers for repeated questions.

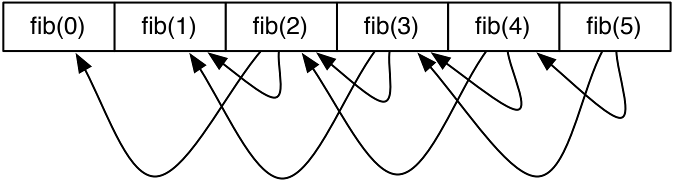
$$\mathrm{fib}(n) = \begin{cases} n & \text{if } n = 0 \text{ or } n = 1 \\ \mathrm{fib}(n-2) + \mathrm{fib}(n-1) & \text{if } n \geq 2 \end{cases}$$



IDEA 2: "DYNAMIC PROGRAMMING"

Compute each value exactly once, in a "clever" order.
Ensure that problems are solved after their subproblems.

$$\mathrm{fib}(n) = \begin{cases} n & \text{if } n = 0 \text{ or } n = 1 \\ \mathrm{fib}(n-2) + \mathrm{fib}(n-1) & \text{if } n \geq 2 \end{cases}$$



Recall: The Knapsack Problem

Suppose we have n kinds of items.

- ✓ They have value (or utility) $v_1, \ldots, v_n$
- ✓ Each has weight (or size or cost) $w_1, \ldots, w_n$.

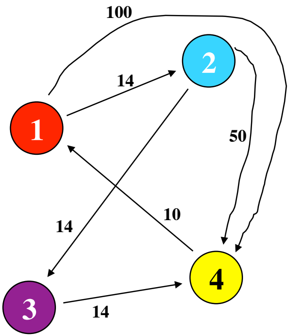
What should we choose, if the maximum total weight is W?

$$\mathrm{optv}(0) = 0$$

$$\mathrm{optv}(W) = \max \begin{cases} \mathrm{optv}(W-1) \\ v_1 + \mathrm{optv}(W - w_1) \\ v_2 + \mathrm{optv}(W - w_2) \\ \vdots \\ v_n + \mathrm{optv}(W - w_n) \end{cases}$$

Floyd-Warshall

An algorithm for finding *all-pairs* shortest paths!



For k = 0, 1, 2, . . ., compute:

What is the shortest path from s *to* d *via nodes* 1..k *only?*

Floyd-Warshall Algorithm

Minimum distance from **`src`** to **`dst`** using intermediate nodes 1..k

`T[k][src][dst]`

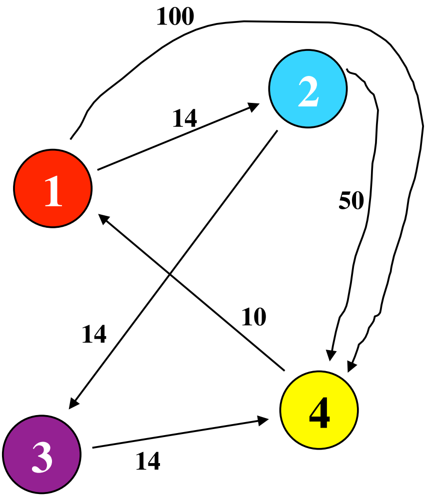
lose k

```
= min { T[k-1][src][dst]

        T[k-1][src][k]  +
        T[k-1][k][dst]
```

use k

Example: k = 0

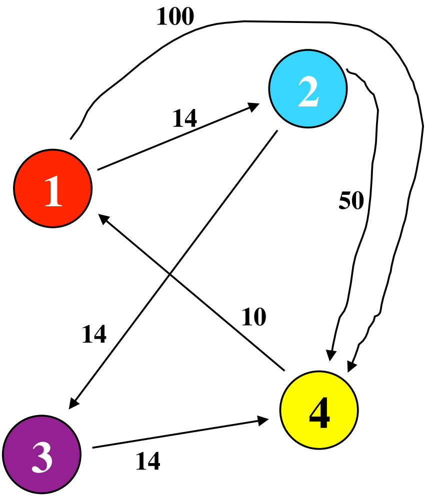


dst
"to"

src
"from"

src \ dst	1	2	3	4
1	0	14	inf	100
2	inf	0	14	50
3	inf	inf	0	14
4	10	inf	inf	0

EXAMPLE: K = 1

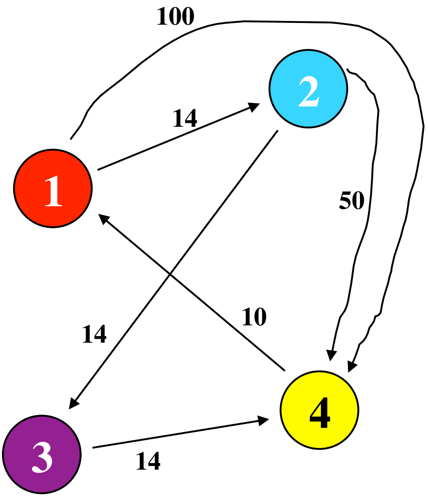


dst "to"

src "from"

	1	2	3	4
1	0	14	inf	100
2	inf	0	14	50
3	inf	inf	0	14
4	10	24	inf	0

EXAMPLE: K = 2

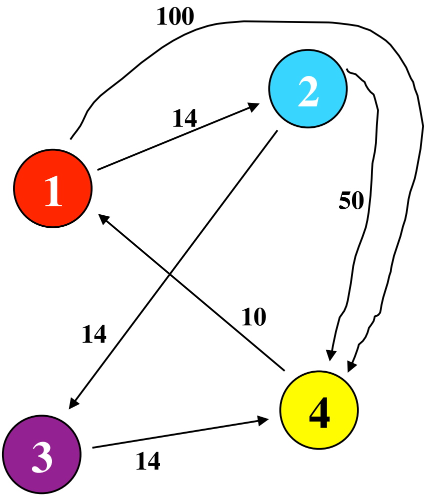


dst "to"

src "from"

src \ dst	1	2	3	4
1	0	14	28	64
2	inf	0	14	50
3	inf	inf	0	14
4	10	24	38	0

Example: k = 2

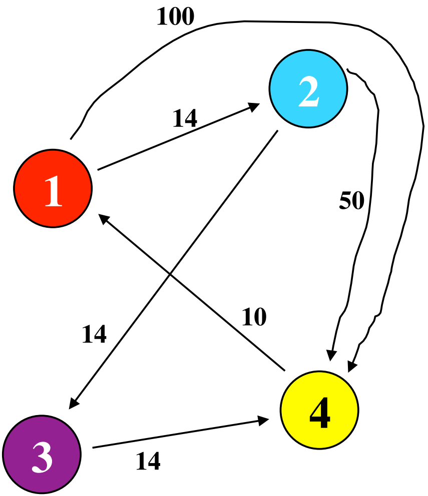


dst
"to"

src
"from"

src \ dst	1	2	3	4
1	0	14	28	42
2	inf	0	14	28
3	inf	inf	0	14
4	10	24	38	0

Example: k = 4 (done)



dst "to"

src "from"

	1	2	3	4
1	0	14	28	42
2	38	0	14	28
3	24	38	0	14
4	10	24	38	0