CS3000: Algorithms & Data
Drew van der Poel

Recitation 3:

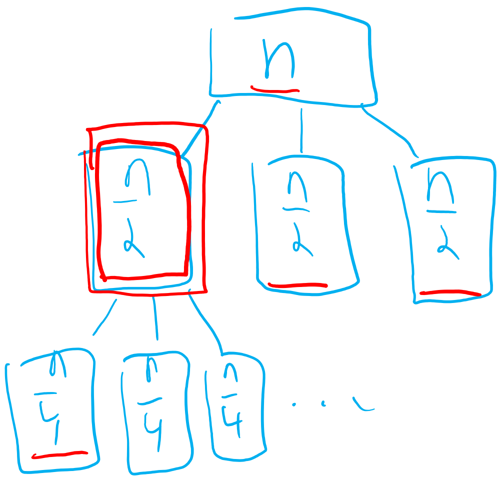
- Recursion trees
- Dynamic Programming

May 24, 2021

P1- Recursion Tree

$$T(n) = 3 \cdot T(n/2) + \boxed{Cn^2}$$
$$T(1) = C$$

level	#
0	1
1	3
2	9
⋮	
i	3^i
$\log_2 n$	



size	total @ level
$n = n/2^0$	Cn^2
$\frac{n}{2} = n/2^1$	$C\left(\frac{n}{2}\right)^2 + C\left(\frac{n}{2}\right)^2 + C\left(\frac{n}{2}\right)^2 = 3C\left(\frac{n}{2}\right)^2 = 3C\frac{n^2}{4}$
$\frac{n}{4} = \frac{n}{2^2}$	$9C\left(\frac{n}{4}\right)^2 = 9C\frac{n^2}{16}$
$n/2^i$	$3^i\, C\left(\frac{n}{2^i}\right)^2 = \left(\frac{3}{4}\right)^i Cn^2$

MT: $a=3, b=2, d=2$

$\frac{3}{2^2} < 1 \rightarrow$ case 3 $\Theta(n^2)$

$$\frac{n}{2^i} = 1 \rightarrow n = 2^i \rightarrow \log_2 n = i$$

geom. seq $r<1$

$$\sum_{i=0}^{\log_2 n} \left(\frac{3}{4}\right)^i Cn^2 = \boxed{Cn^2 \sum_{i=0}^{\log_2 n} \left(\frac{3}{4}\right)^i} = Cn^2C' = \Theta(n^2)$$

Problem 2

In the Subset Sum problem, we are given a set $A = \{a_1, a_2, \ldots, a_n\}$ of n integers and another integer b, and are asked to determine if there exists a subset of A whose sum is exactly b. So, an algorithm for Subset Sum takes as input A and b and returns **True** if there exists a subset of A whose sum is exactly b, and **False** otherwise.

(a) Suppose you are given an algorithm to solve the Knapsack problem. Show how the Subset Sum problem can be solved by building an instance of the Knapsack problem and then using the Knapsack algorithm.

$A = \{1, 2, 4\}, b = 2$ $\longrightarrow$ $v_1 = w_1 = 1$, $v_2 = w_2 = 2$, $v_3 = w_3 = 4$ $\quad T = 2 = b$

INPUTS

SS

n integers

b - target

$\longrightarrow$

Knapsack

n' items w/ weight + value

T - capacity

n items

$a_i \rightarrow w_i = v_i = a_i$

$\forall\, i; 1 \le i \le n$

$T = b$

- if opt. sum has value b $\rightarrow$ TRUE for SS
- else $\Rightarrow$ False for SS

(b) Here, we consider a more direct dynamic programming algorithm. Let SUBSETSUM$[i,s]$ be **True** if there is a subset of $\{a_1,\ldots,a_i\}$ whose sum is s, and **False** otherwise. Note that the Subset Sum problem is asking to find SUBSETSUM$[n,b]$. Show that for any $i>1$ and $s>0$, SUBSETSUM$[i,s]$ can be determined from the values of SUBSETSUM$[i-1,\cdot]$. Using this, establish a recurrence relation for SUBSETSUM$[i,s]$. Remember to include base cases.

1st i items

target

Case 1 - ith item excluded

Case 2 - ith item included

$$SS[i, s] = SS[i-1, s] \quad \text{OR} \quad SS[i-1, s-a_i]$$

Base Cases

$SS[0, s>0] = \text{False}$

$SS[i, 0] = \text{True}$

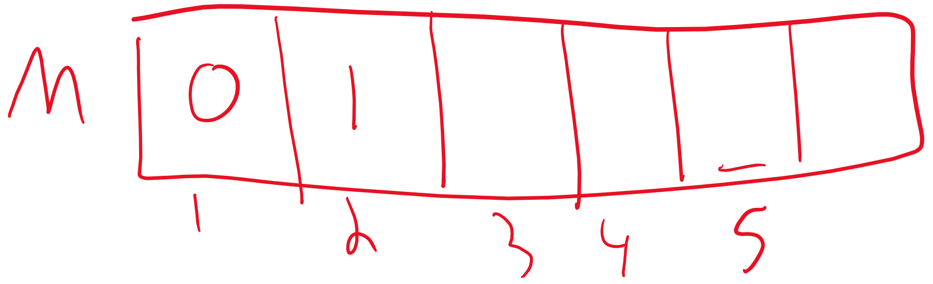
top-down

(c) Convert the recurrence of part (b) to a bottom-up dynamic programming algorithm. Analyze its running time.

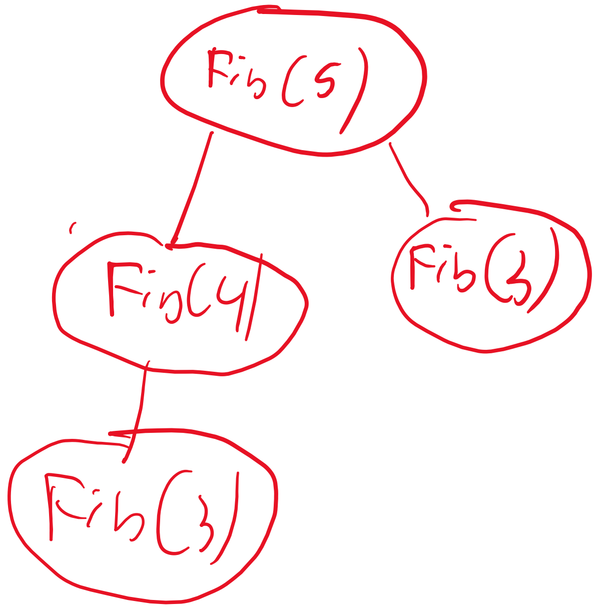
```
Find OPT
        For i from 0 to n:
O(n)        M[i, 0] = TRUE
O(b)    For s from 1 to b:
            M[0, s] = FALSE
        For i from 1 to n:
            For s from 1 to b:
            → M[i,s] = M[i-1, s] OR M[i-1, s-a_i]
        Return M[n, b]
```

Runtime: $O(nb)$

Fib - top down for n = 5



Fib (5)



Problem 3

Problem 2. *Dynamic Programming*

The dark lord Sauron loves to destroy the kingdoms of Middle Earth. But he just can't catch a break, and is always eventually defeated. After a defeat, he requires three epochs to rebuild his strength and once again rise to destroy the kingdoms of Middle Earth. In this problem, you will help Sauron decide in which epochs to rise and destroy the kingdoms of Middle Earth.

The input to the algorithm consists of the numbers $x_1, \ldots, x_n$ representing the number of kingdoms in each epoch. If Sauron rises in epoch i then he will destroy all x_i kingdoms, but will not be able to rise again during epochs $i+1, i+2$, or $i+3$. We call a set $S \subseteq \{1, \ldots, n\}$ of epochs *valid* if it satisfies this constraint that $|i-j| \geq 4$ for all $i, j \in S$, and its *value* is $\sum_{i \in S} x_i$. You will design an algorithm that outputs a valid set of epochs with the maximum possible value.

$x_1\ x_2\ x_3 \ldots$

Example: Suppose there are $(1, 7, 8, 2, 6, 3)$ kingdoms of Middle Earth in epochs $1, \ldots, 6$. Then the optimal set of epochs for Sauron to rise up and destroy the kingdoms of Middle Earth is $S = \{2, 6\}$, during which he destroys 10 kingdoms, 7 in the 2nd epoch and 3 in the 6th epoch.

Using DP…

*** describe the set of subproblems you consider**

*** give a recurrence expressing the solution to each subproblem in terms of the solution to smaller subproblems**

***sketch pseudocode of your algorithm & give the runtime**

***describe how you would recover the solution (epochs) if asked**

Subproblems:

Recurrence: