CE2101/ CZ2101: Algorithm Design and Analysis

Week 9

Liu Ziwei

Outline

- Quiz: Week 10 & Week 11
- Substitution Method
- Linear Homogeneous Recurrence Relation
- Dynamic Programming
- Extended Topics (not covered in the exam)

Please feel free to interrupt me if you have any questions :)

Substitution Method

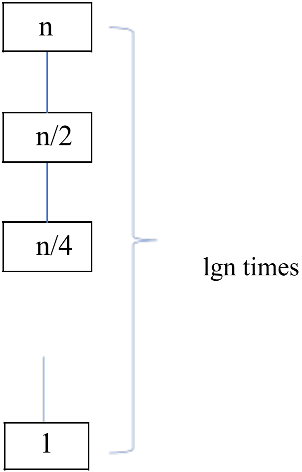
Solving Recurrences – Substitution method

References for guess:

Binary search:

W(n) = W(n/2) + 1,

W(1) = 1



W(n) =
O(lgn)

Solving Recurrences – Substitution method

References for guess:

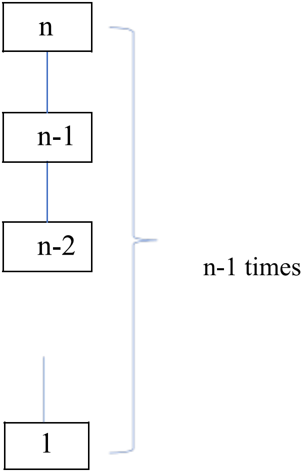
Insertion sort (best case)

$W(n) = W(n-1) + \quad 1,$

$W(1) = 0$

$W(n) =$

$O(n)$



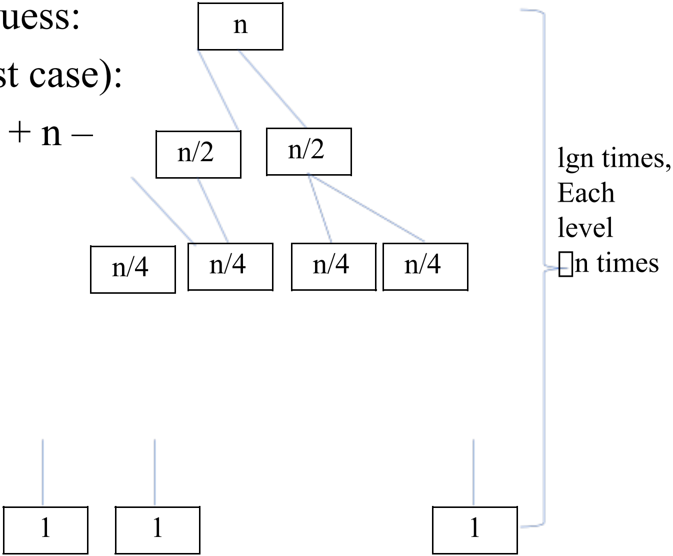
Solving Recurrences – Substitution method

References for guess:

MergeSort (worst case):

$W(n) = 2W(n/2) + n - 1$, $W(1) = 0$

$W(n) = O(n\lg n)$

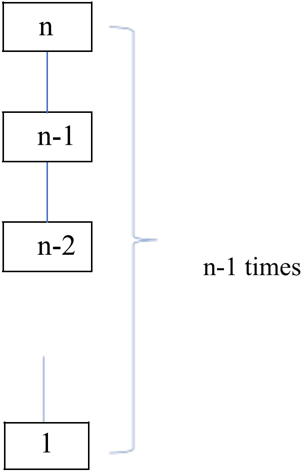


Solving Recurrences – Substitution method

References for guess:

QuickSort (worst case): $W(n) = W(n-1) + n - 1$, $W(1) = 0$

$W(n) = O(n2)$



Solving Recurrences – Substitution method

References for guess:

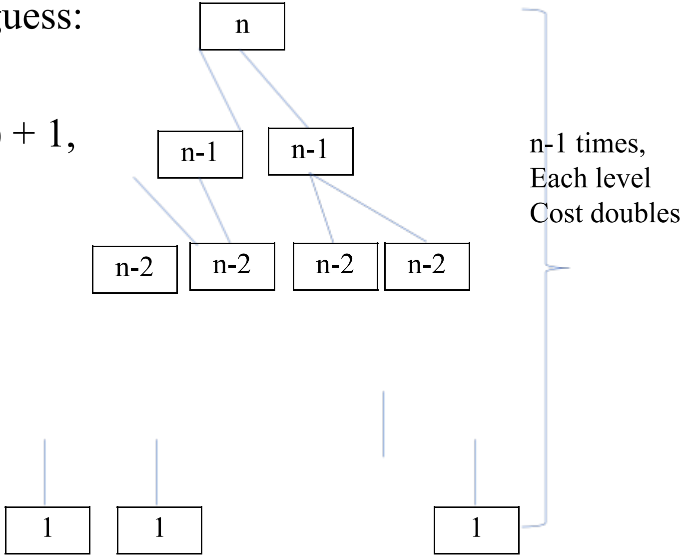
Tower of Hanoi

$W(n) = 2W(n-1) + 1,$

$W(1) = 0$

$W(n) =$

$O(2n)$



Linear Homogeneous Recurrence Relation

We consider a linear homogeneous recurrence relation of degree 2

$a_n = Aa_{n-1} + Ba_{n-2}$ for all $n \geq 2$

where A and B are real constants

The **characteristic equation**

$$t^2 - At - B = 0$$

may have

1) two distinct roots

2) a single root

Theorem 1 (Distinct Roots Theorem)

Suppose a sequence $a_0, a_1, a_2, \ldots$ satisfies a recurrence relation

$a_n = Aa_{n-1} + Ba_{n-2}$ *for all* $n \geq 2$

where A and B are real constants and $B \neq 0$. If the characteristic equation

$t^2 - At - B = 0$

has two distinct roots r and s, then $a_0, a_1, a_2, \ldots$ is given by the explicit formula

$$\boldsymbol{a_n = Cr^n + Ds^n}$$

where C and D are determined by the values of a_0 and a_1.

Theorem 2 (Single-Root Theorem)

Suppose a sequence *a0, a1, a2, ….* satisfies a recurrence relation

an = Aan-l + Ban-2 for all n $\geq$ *2*

where A and B are real constants and B $\neq$ 0. If the characteristic equation

t2 - At – B = 0

has a single (real) root, then *a0, a1, a2, ….* is given by the explicit formula

an = Crn + Dnrn

where C and D are determined by the values of *a0* and any other known value of the sequence.

Dynamic Programming

Sudoku

	C1	C2	C3	C4	C5	C6	C7	C8	C9
R1	9	6					7		8
R2	8					4	3		
R3	1			5					
R4							1	7	6
R5	2				9	3			5
R6	7		8						
R7			7		3	2		4	
R8	3	8	2	1		5	6		
R9		4	1			9	5	2	

Dynamic Programming

Consider the Virahanka number $V(n)$ defined by the following equations:

$V(n) = 1$, when $n = 0$ or 1;

$V(n) = V(n-1) + V(\lfloor n/2 \rfloor)$, when $n>1$;

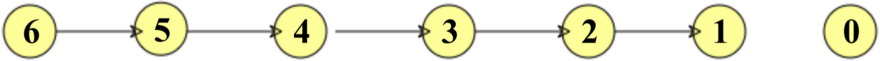
(here $\lfloor n/2 \rfloor$ is the floor function for $n/2$. E.g. $\lfloor 3/2 \rfloor$ $=$ $\lfloor 2/2 \rfloor =1$).

Dynamic programming (Bottom Up)

<u>Subproblem graphs</u>

For a recursive algorithm A, the subproblem graph for A is a directed graph whose vertices are the instances for this problem. The directed edges (I, J) for all pairs that indicate: if A is invoked on problem I, it makes a recursive call directly on instance J.

E.g. the subproblem graph for fib(6):



Dynamic programming (Bottom Up)

1. Formulate the problem P in terms of smaller versions of the problem (recursively), say, Q1, Q2, …
2. Turn this formulation into a recursive function to solve problem P
3. Draw the subproblem graph and find the dependencies among subproblems
4. Use a dictionary to store solutions to subproblems
5. In the iterative function to solve P

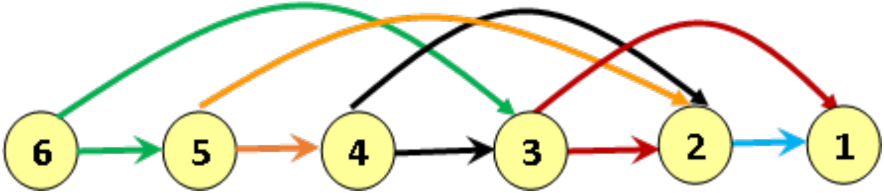
- compute the solutions of subproblems of a problem first
- The solution to P is computed based on the solutions to its subproblems and is stored into the dictionary

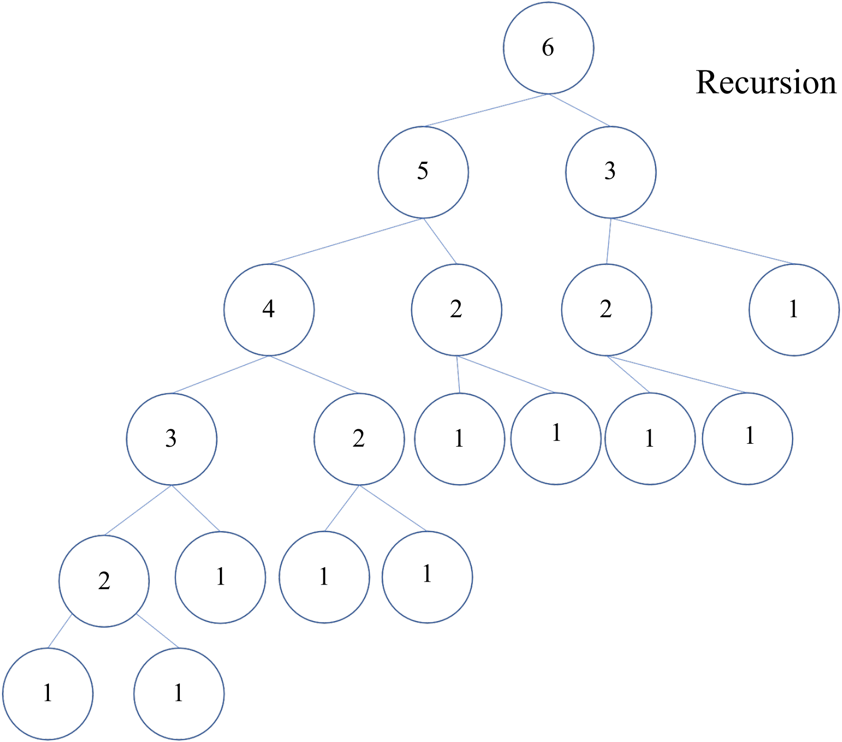
Dynamic Programming

$V(n) = 1$, when $n = 0$ or 1;

$V(n) = V(n-1) + V(\lfloor n/2 \rfloor)$, when $n>1$;

(i) Draw the subproblem graph for $V(6)$.





Recursion tree

Dynamic Programming

$V(n) = 1$, when $n = 0$ or 1;

$V(n) = V(n-1) + V(\lfloor n/2 \rfloor)$, when $n>1$;

(ii)Give dynamic programming algorithm to compute V(n) using the bottom up approach.

```
v[0] = v[1] = 1
For (j= 2 to n)
    v[j] = v[j-1] +
v[j/2]; return v[n];
```

Dynamic Programming

$V(n) = 1$, when $n = 0$ or 1;

$V(n) = V(n-1) + V(\lfloor n/2 \rfloor)$, when $n>1$;

(iii) Give dynamic programming algorithm to compute $V(n)$ using the top down approach.

Dynamic programming (Top Down)

1. Formulate the problem P in terms of smaller versions of the problem (recursively), say, Q1, Q2, …
2. Turn this formulation into a recursive function to solve problem P
3. Use a dictionary to store solutions to subproblems
4. In the recursive function to solve P

- Before any recursive call, say on subproblem Qi, check the dictionary to see if a solution for Qi has been stored
    - If no solution has been stored, make the recursive call
    - Otherwise, retrieve the stored solution
- Just before returning the solution for P, store the solution in the dictionary - memorization

```
// v array initialized to all -1 DP_V(n)
{
If (n == 0 or n == 1) { v[n] = 1; return 1 }
If (v[n-1] == -1) v1 = DP_V(n-1) else v1 = v[n-
1]; If (v[n/2] == -1) v2 = DP_V(n/2) else v2 =
v[n/2]; v[n] = v1 + v2;
Return v[n];
}
```

Thanks!

Q & A