DESIGN AND ANALYSIS OF ALGORITHMS (DAA)

RECAP: RECURSIVE FUNCTION

COIN SELECTION PROBLEM AND FURTHER ANALYSIS

Course Instructor: Dr. Shreya Ghosh

Recorded Presentation (17th Jan)

Recap: Fibonacci Numbers

$$f(n) = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ f(n-1)+f(n-2), & \text{if } n>1 \end{cases}$$

$$f(n) = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$$

*Golden Ratio

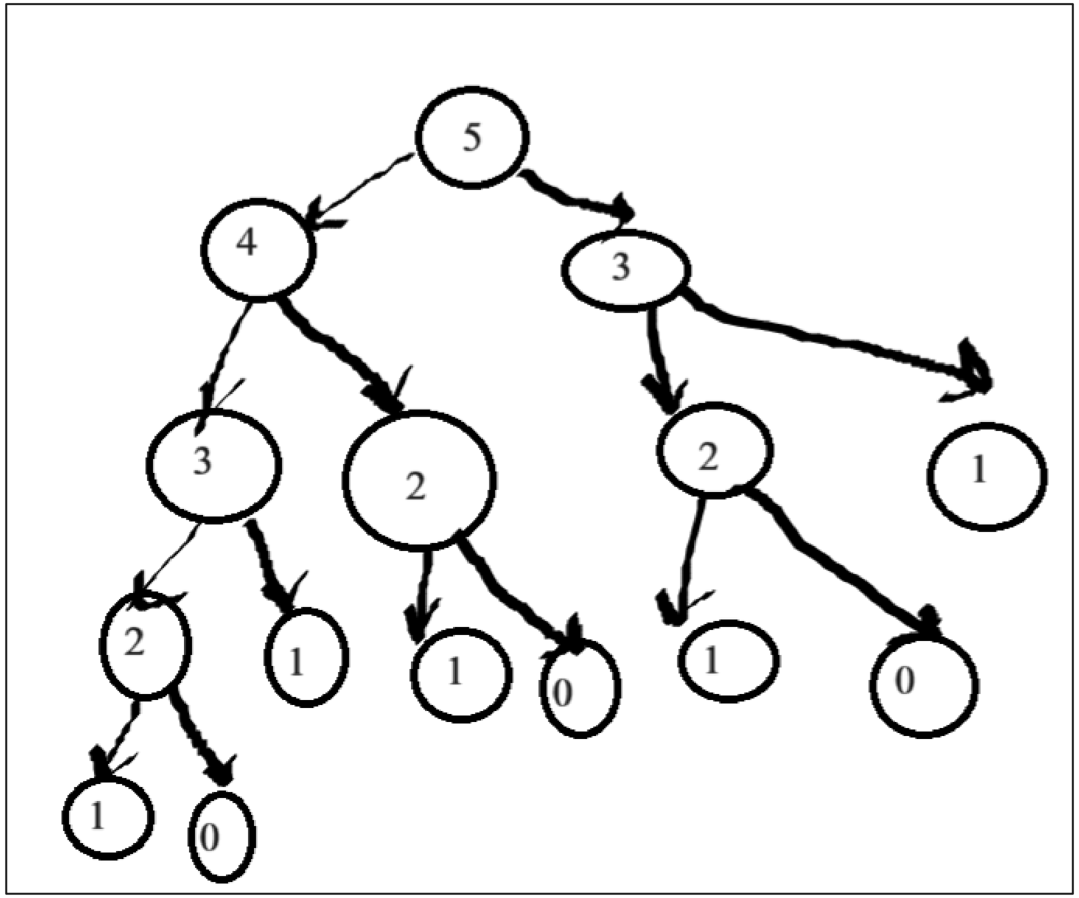
```
Fib(n)
{
        if (n<=0) return(0);
        if(n=1)   return(1);
        m=fib(n-1)+fib(n-2)
        return (m)
}
```

0,1,1,2,3,5,8,....

$$T(n) = 0 \text{ if}(n<=1)$$
$$= T(n-1)+T(n-2)+1$$
$$= f(n+1) - 1$$

Fibonacci Numbers as a Natural Phenomenon: https://science.howstuffworks.com/math-concepts/fibonacci-nature.htm

Fibonacci Sequence: Analysing the Recursion Structure



Identical Subproblems

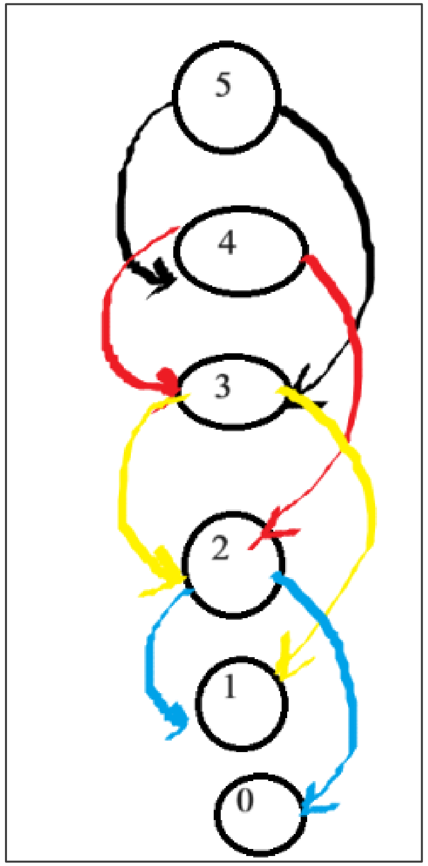
We would like to compute the value of f(n) only once for every n and reuse the same

Data Storage: To store the required past computations

Memoization

*Memoization is a computer science technique that speeds up the execution of functions by storing the results of function calls and returning them when the same inputs occur again

Fibonacci Sequence: Analysing the Recursion Structure



NO Identical Subproblems

We would like to compute the value of f(n) only once for every n and reuse the same

Data Storage: To store the required past computations

Memoization

*Memoization is a computer science technique that speeds up the execution of functions by storing the results of function calls and returning them when the same inputs occur again

$$T(n) = O(n)$$

Memoization

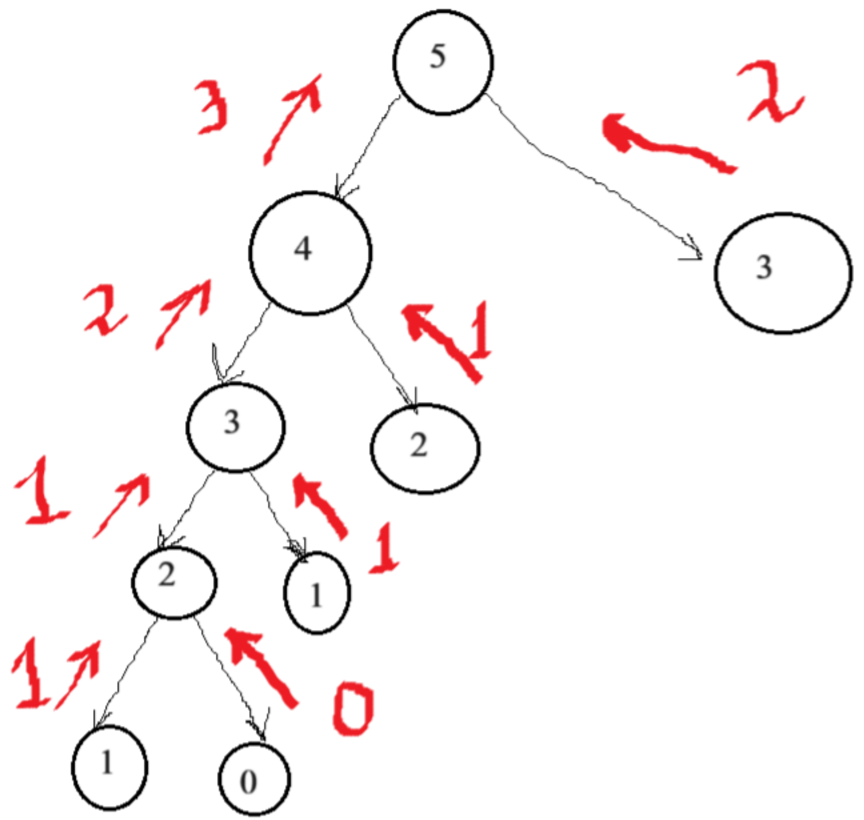
FIB[ ] FIB[0]=0 FIB[1]=1

<u>Top-down algorithm</u>

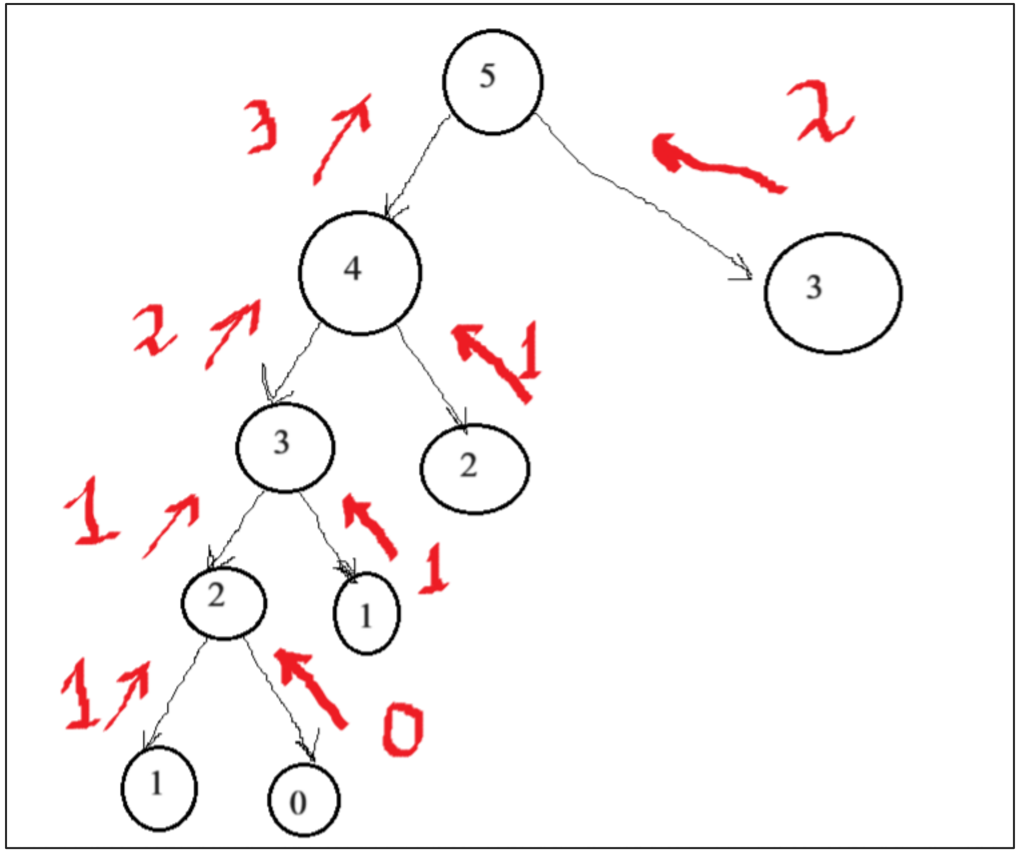
Done[ ] Done[0]=1 Done[1]=1

All other are 0

```
Fib2(n)
{
        if(Done[n]=1) return (FIB[n]);
        m= Fib2(n-1)+Fib2(n-2);
        Done[n]=1;
        Fib[n]=m;
        return(m);
}
```



Memoization



Done[0]=1 FIB[0]=0
Done[1]=1 FIB[1]=1

	0	1	2	3	4	5
Done	1	1	1	1	1	1
FIB	0	1	1	2	3	5

$$T(n)=O(n)$$
$$S(n) = O(n)$$

Finalizing the Algorithm

FIB[0]=0
FIB[1]=1

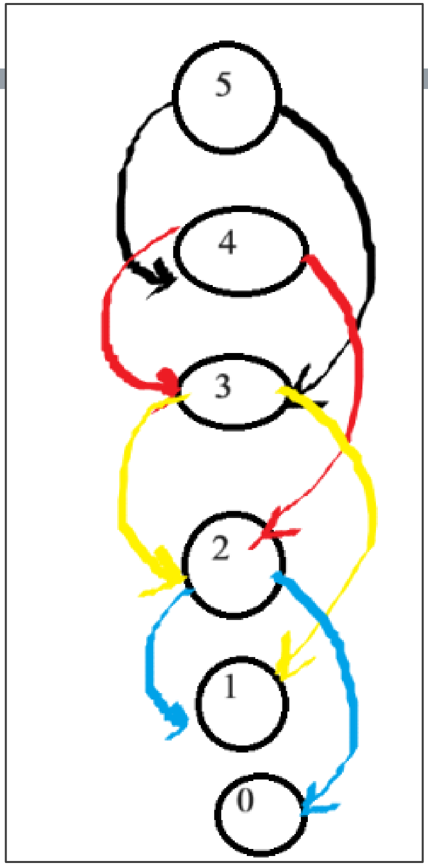
```
Fib3(n)
{
        for i=2 to n do
                FIB[i]=FIB[i-1]+FIB[i-2]
}
```

0	1	2	3	4
0	1	1	2	3

Bottom Up evaluation

Finalizing the Algorithm

```
Fib4(n)
{
        x1 =1 // FIB[1]
        x2=0  // FIB[0]
        for i=2 to n do
        {
                m=x1+x2;
                x1=x2
                x2=m
        }
    return (m)
}
```



DIY: Using TAIL RECURSION

Variations

1. $f(n) = f(n-1) + f(n-157)$
2. $f(n) = f(n-1) + f\left(n - \frac{n}{2}\right)$
3. $f(n) = f(n+1) + f(n+3)$ if n is even
   $f\left(\frac{n-1}{2}\right)$ if n is odd
4. $f(n) = f(g(n)) + f(h(n))$
   $\hookrightarrow$ possibility of cyclic dependencies

DIY questions:

- How can you evaluate it?
- How many memory locations will be required?
- How do the iterations look like?
- What happens in the case of cyclic dependency?

DESIGN AND ANALYSIS OF ALGORITHMS (DAA)

RECAP: RECURSIVE FUNCTION

COIN SELECTION PROBLEM AND FURTHER ANALYSIS

Course Instructor: Dr. Shreya Ghosh

Recorded Presentation (17th Jan)

Coin Selection Problem

Given a set C of n coins having denomination values $\{c_1, c_2, \dots, c_n\}$ and a desired final value of V, find the minimum number of coins to be chosen from C to get an exact value of V from the sum of denominations of the chosen subset

example: $C = \{8, 6, 5, 2, 1\}$, $V = 11$

$S_1 = \{8, 2, 1\}$, $S_2 = \{6, 5\}$

↑ minimum

Coin Selection Problem

Given a set C of n coins having denomination values $\{c_1, c_2, \ldots, c_n\}$ and a desired final value of V, find the minimum number of coins to be chosen from C to get an exact value of V from the sum of denominations of the chosen subset

example: $C = \{8, 6, 5, 2, 1\}$, $V = 11$

$S_1 = \{8, 2, 1\}$, $S_2 = \{6, 5\}$

↑ minimum

Coins (S, T, x, z, n)

S: set of coins selected till now

T: remaining set of coins from which we can select

x: value of set S

z: remaining value desired to be chosen from T

n: The number of coins selected

$\boxed{\text{Coins}(NULL, C, 0, V, 0)}$

→ Base condition

→ Recursive condition

First Recursive Definition

$\langle P, d \rangle = coins(S, T, x, z, n)$

$\{$ Let $S = \{s_1, s_2, \dots, s_n\}$

$T = \{t_1, t_2, \dots, t_m\}$

<u>BASE CONDITIONS</u>

If $(z = 0)$ return $(\langle S, n \rangle)$

If $(z < 0)$ return $(\langle NULL, \infty \rangle)$

If $(T = NULL)$ return $(\langle NULL, \infty \rangle)$

$P_{min} = NULL$

$min = \infty$

First Recursive Definition

$\langle P, d \rangle = coins(S, T, x, z, n)$

{ Let $S = \{s_1, s_2, \ldots, s_n\}$

$T = \{t_1, t_2, \ldots, t_m\}$

BASE CONDITIONS

If $(z = 0)$ return $(\langle S, n \rangle)$

If $(z < 0)$ return $(\langle NULL, \alpha \rangle)$

If $(T = NULL)$ return $(\langle NULL, \alpha \rangle)$

$P_{min} = NULL$

$min = \alpha$

RECURSIVE CONDITION

for $(i = 1$ to $m)$ do

{ $W = S + \{t_i\}$

$U = T - \{t_i\}$

$\langle P', d' \rangle = coins(W, U, x + t_i, z - t_i, n+1)$

if $(d' < min)$

{ $min = d'$

$P_{min} = P'$

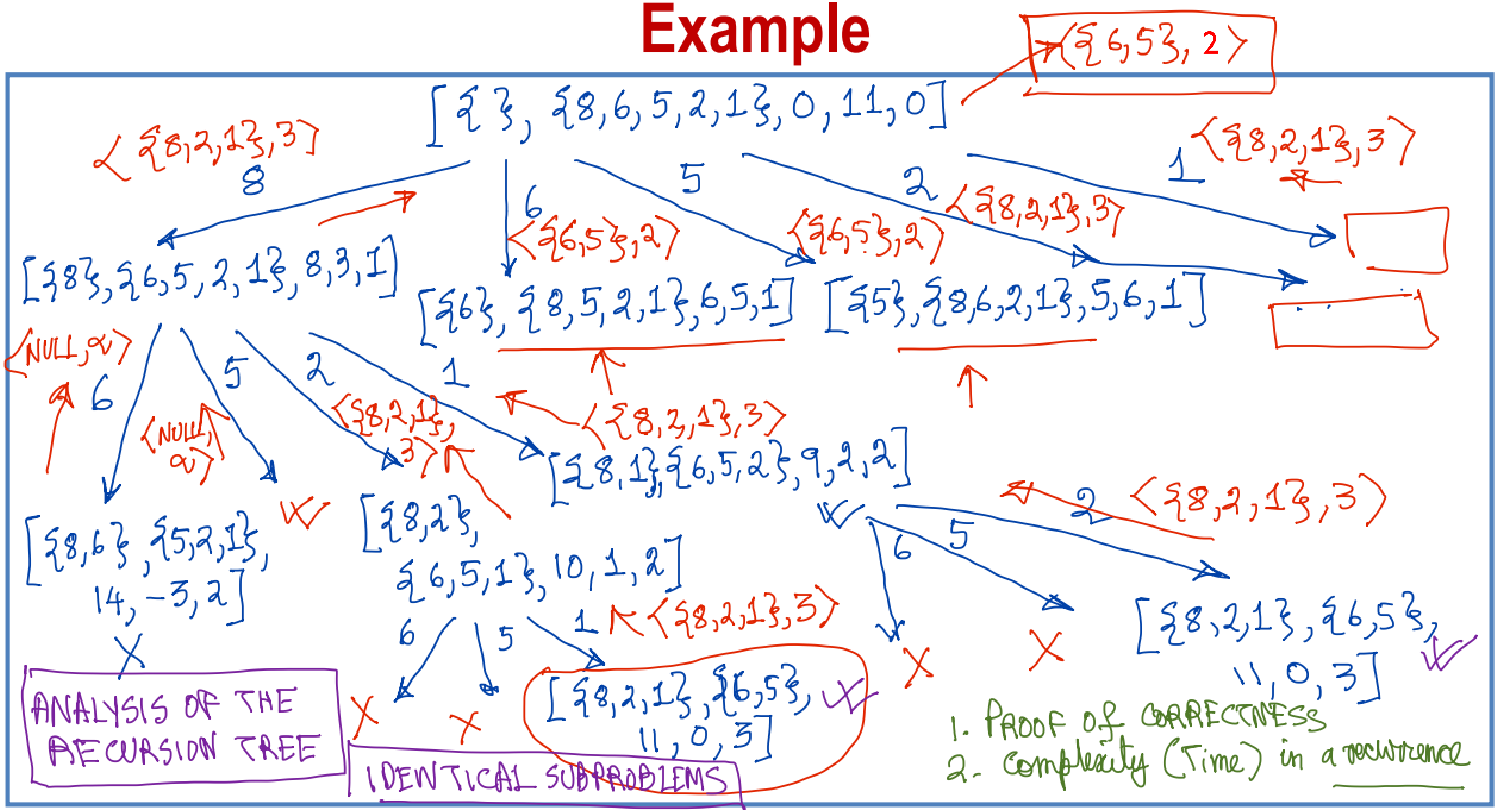
}

}

return $(\langle P_{min}, min \rangle)$

}

Example



<{6,5}, 2>
[{ }, {8,6,5,2,1}, 0, 11, 0]
<{8,2,1},3>
8
6
5
2
1
<{8,2,1},3>
[{8},{6,5,2,1}, 8,3,1]
[{6}, {8,5,2,1},6,5,1]
[{5},{8,6,2,1},5,6,1]
<{6,5},2>
<{6,5},2>
<{8,2,1},3>
<NULL, ∞>
<NULL, ∞>
<{8,2,1},3>
<{8,2,1},3>
[{8,1},{6,5,2},9,2,2]
[{8,6},{5,2,1}, 14,-3,2]
[{8,2}, {6,5,1}, 10,1,2]
<{8,2,1},3>
<{8,2,1},3>
[{8,2,1},{6,5}, 11,0,3]
[{8,2,1},{6,5}, 11,0,3]
ANALYSIS OF THE RECURSION TREE
IDENTICAL SUBPROBLEMS
1. PROOF OF CORRECTNESS
2. Complexity (Time) in a recurrence

Improved Recursive Definition

Instead of

$$U = T - \{t_i\}$$

we do the following

$$U = T - \underline{\{t_1, t_2, \dots, t_i\}}$$

Improved Recursive Definition

Instead of

$$U = T - \{t_i\}$$

we do the following

$$U = T - \{t_1, t_2, \ldots, t_i\}$$

$$[\{\}, \{8, 6, 5, 2, 1\}, 0, 11, 0]$$

8: $[\{8\}, \{6, 5, 2, 1\}, 8, 3, 1]$

6: $[\{6\}, \{5, 2, 1\}, 6, 5, 1]$

5: $[\{5\}, \{2, 1\}, 5, 6, 1]$

Identical subproblems that were generated earlier will not be generated now.

1. PROOF OF CORRECTNESS
2. TIME COMPLEXITY BASED ON Recurrence Relation
3. ANALYSIS OF RECURSION STRUCTURE

Alternative Recursive Definition

$\langle P, d\rangle = \text{coins2}(S, T, x, z, n)$

$\{$ Let $S = \{s_1, s_2, \dots, s_n\}$

$T = \{t_1, t_2, \dots, t_m\}$

BASE CONDITIONS

If $(z=0)$ return $(\langle S, n\rangle)$

If $(z<0)$ return $(\langle NULL, \infty\rangle)$

If $(T = NULL)$ return $(\langle NULL, \infty\rangle)$

$P_{min} = NULL$

$min = \infty$

Recursive condition

(Inclusion - Exclusion Principle)

$\langle P_1, d_1\rangle = \text{coins2}(S+t_1, T-t_1, x+t_1, z-t_1, n+1)$

$\langle P_2, d_2\rangle = \text{coins2}(S, T-t_1, x, z, n)$

If $(d_1 \le d_2)$

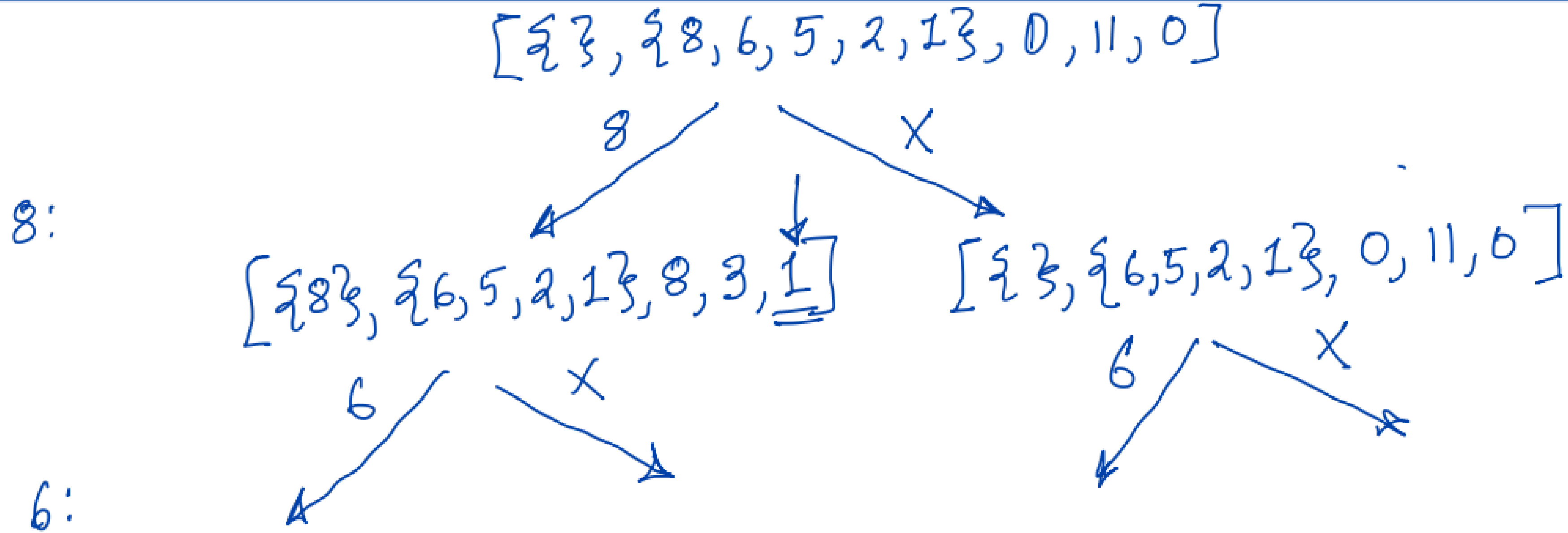
$\{P_{min} = P_1;\ min = d_1\}$

else $\{P_{min} = P_2;\ min = d_2\}$

return $(\langle P_{min}, min\rangle)$

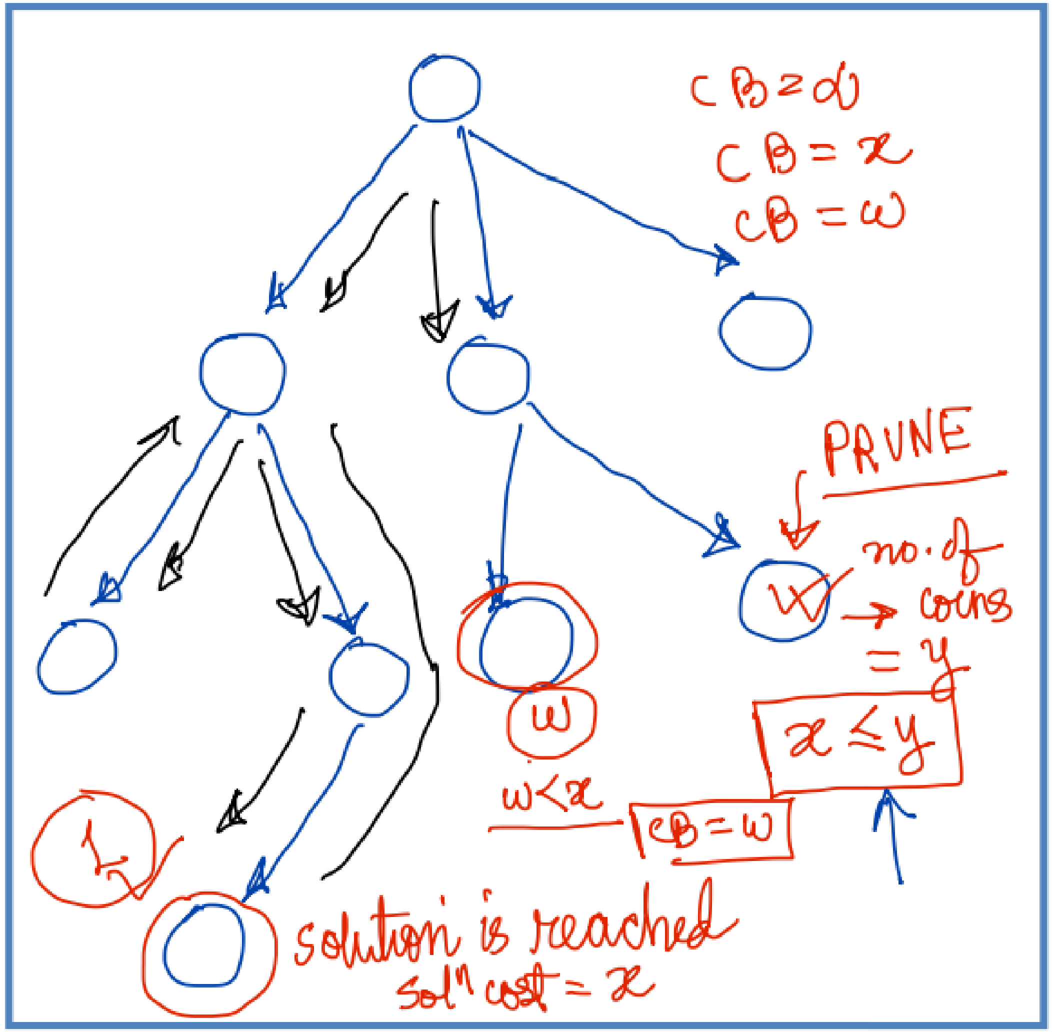
$\}$

Example



1. Inductive Proof
2. Time Recurrence
3. Identical Subproblems

Traversal and Potential Pruning



Maintain a global current best

$CB = \infty$ (initially)

Recursion is evaluated in a depth-first manner

BASE CONDITIONS are Revised for Pruning

If $(Z=0)$ { If $(n < CB)$, $CB = n$ [update the current best]

return($\langle S, n \rangle$)

}

If $(Z<0)$ return($\langle NULL, \infty \rangle$)

If $(T = NULL)$ return($\langle NULL, \infty \rangle$)

$\boxed{\text{If } (n \geq CB) \text{ return}(\langle NULL, \infty \rangle)}$

PRUNING

Special Case

$C = \{16, 8, 4, 2, 1\}$ $\{2^i\}$

If $V = 25$

16 8 X 4 X 2 X 1 X

choice for 16 will be part of optimal solution

$[\{16\}, \{8, 4, 2, 1\}\ 16, 9, 1]$

8 1 X 2 X 1 X

$[\{16, 8\}\ \{4, 2, 1\}, 24, 1, 2]$

4 X 2 X 1 SOLUTION

We can make a SINGLE choice from the various recursive sub-problems.

$\{100, 50, 25, 20, 10, 5, 3, 2, 1\}$

Summary

1. Initial Solution
2. Analyze the Recursion
   (a) Balancing the split [D&C]
   (b) Identical Sub-problems (Memoization) [DP]
   (c) Choice (Greedy) from the subproblems upfront [G]
   (d) Traversal or Evaluation of the Recursion allows for pruning or pre-emption based on solutions already found. [BB]

3. Proof of correctness
4. Analysis of Complexity [Recurrence Eqns]
5. Data Structures [Asymptotic Analysis]

Problems we have examined

1. Max, Max-Min, Max1-Max2
2. FIB
3. Coins