Dynamic Programming

[6, 9, 8, 1, 1, 5]

[10, 2, 3, 14]

10 → 10

2 → 10

3 → 13

14 → 24 (24, 13)

0 → 24

6 → 6

9 → 9

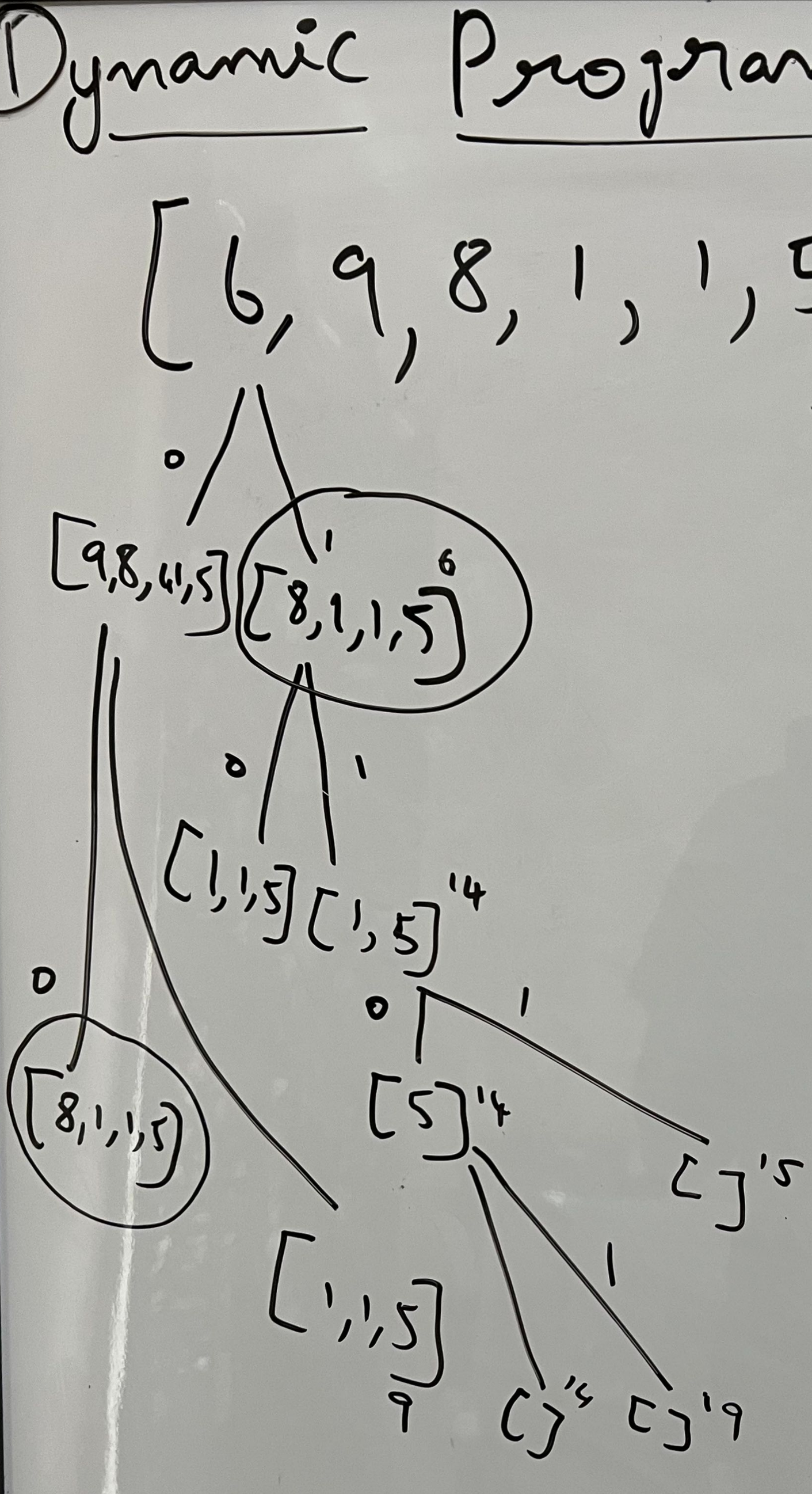
8 → 14 → max(8+6, 9)

1 → 14 → max(1+9, 14)

1 → 15 → max(14+1, 14)

5 → 19 → max(5+14, 15)

Dynamic Programming



Dynamic Programming

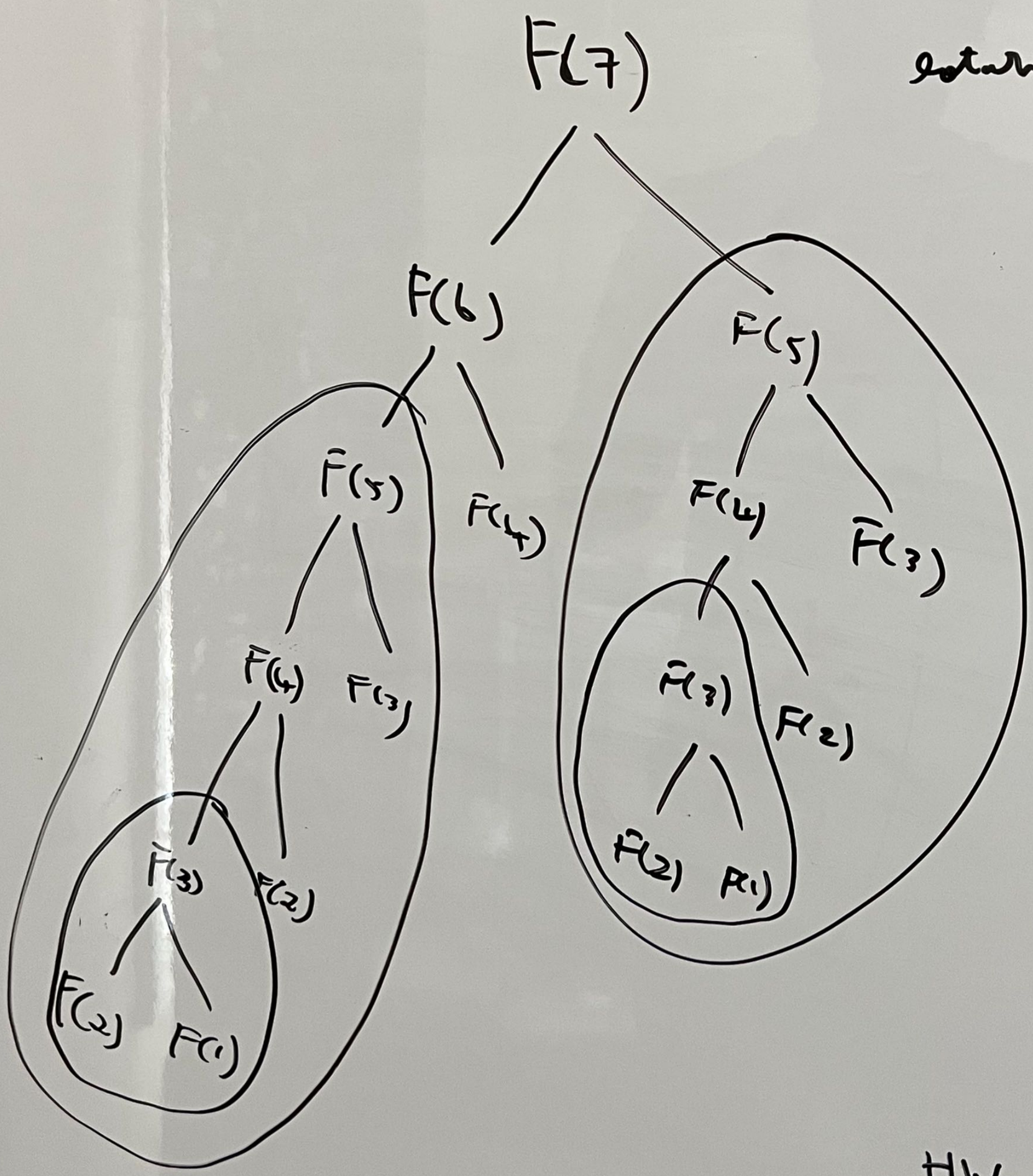
- optimizing Repeated Subproblems

Fib(7)

F(1), F(2)

Fib(n)

return Fib(n-1) + Fib(n-2)



HW

1) 01 KNAPSACK PROBLEM

Subarray sum equals K

K=7

[3, 4, 7, 2, -3, 1, 4, 2, 0, 1]

0, 3, 7, 14, 16, 13, 14, 18, 20, 20, 21

[3, 4]

[7]

[7, 2, -3, 1]

Count = 1 + 1 + 1 + 1 + 1 + 2

C = rsum - K

{ 0 : 1
3 : 1
7 : 1
14 : 2
16 : 1
13 : 1
18 : 1
20 : 2
21 : 1 }

TC: O(n)

SC: O(n)

Subarray sum equals K

K=7

[3,4,7,2,-3,1,4,2,0,1]

0, 3, 7, 14, 16, 13, 14, 18, 20, 20, 21

[3,4]

[7]

[7,2,-3,1]

Count = 1+1+1+1+1
+2

c = rsum − K

0	1
3	1
7	1
14	2
16	1
13	1
18	1
20	2
21	1

TC: O(n)

SC: O(n)