DSA II - L-8 - 16 oct

Greedy Approach/Strategy

Coin change

ex1

Change Amount = 52727

bill = 47273

pay = 100000

coin/notes = [1000, ~~1000~~, 500, 200, 100, 50, 20, 10, 5, 2, 1]

minimum number of coins = 57

Exm-2

Amount = 30

coins = [25, 10, 1]

Current amount = 30

25 X 1 = 25

" " = 5

1 X 5 = 5

RADIANT PHARMACEUTICALS

" " = 0

Total coins = 6

[minimization ~~maximization~~ ---
--- → Greedy]

optimal এর ~~থেকে~~ times
[Greedy ১.৭ এর থেকে worst solution
~~a~~ যেয়না

# amount = 127
coins = [50, 1, 20, 5, 10, 17]

→ sorted coins descending order
= [50, 20, 17, 10, 5, 1]

C.A. = 127
50 × 2 = 100
R.A = 27

C.A = Current Amount
R.A. = Remaining

$20 \times 1 = 20$

R.A $= 7$

$5 \times 1 = 5$

R.A. $= 2$

$2 \times 1 = 2$

0

Total coins ₹

optimal soln. $= 4$

$[4, 4 \times 2.7]$

$= [4, 10.8]$

$\therefore 4 \leq 6 \leq 10.8$

RADIANT
PHARMACEUTICALS

Activity Selection (maximization)

[always optimal solution না
Greedy না ~~এ~~ case-এ.]

Activity	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
Start	1	0	1	4	2	5	3	4
Finish	3	4	2	6	9	8	5	8

→ sort the tasks based on-

- Start time
- Finish " * (optimal solution)
- Duration

	S	F
T_1	1	8
T_2	4	7
T_3	2	4
T_4	3	9

sort → start

	S	F	
T_1	1	8	
T_3	2	4	1 টা
T_4	3	9	
T_2	4	7	

sort → Finish

	S	F
T_3	2	4
T_2	4	7
T_1	1	8
T_4	3	9

2 Gr

sort→Finish time

Activity	a_3	a_1	a_2	a_7	a_4	a_8	a_6	a_5
Start	1	1	0	3	4	4	5	2
Finish	2	3	4	5	6	8	8	9

Picked = 1 + 1 = 2 + 1 = 3

tasks = $[a_3, a_7, a_6]$

availability = ~~2~~ ~~5~~ 8

RADIANT PHARMACEUTICALS

```
struct Task {
    int start;
    int finish;
};

int activitySelection (Task tasks[], int n){
    sort (tasks); // based on finish time
    int picked = 1
    int availability = tasks [0]. finish
    for(int i=1; i<n; i++){
        if (tasks[i].stoort >= availabity){
            picked++;
            availability = tasks[i].finish;
        }
    }
    return picked;
}
```

sort - এর optimal time complexity

$O(n \log n)$

DSA II - L - 10 - 21 oct

Greedy -

knapsack

↳ 0/1 knapsack

↳ fractional "

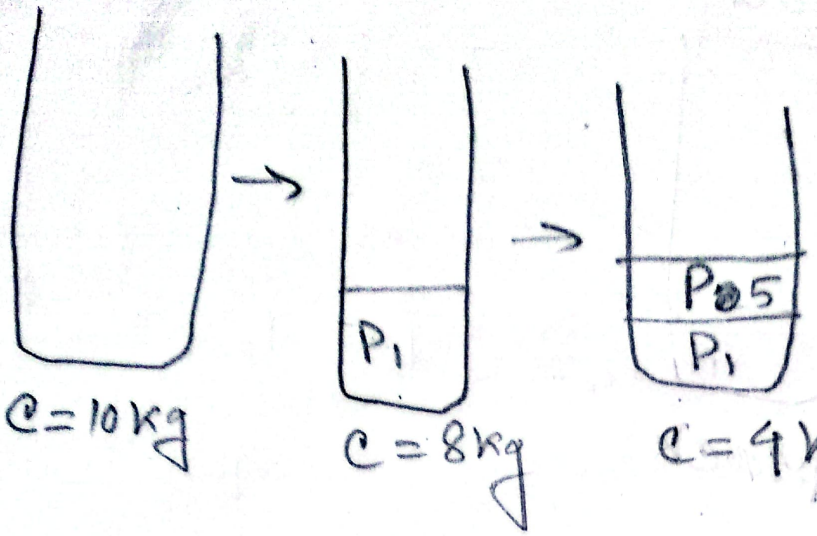
0/1 knapsack

	Product	weight	value	value/weight
1 ←	P_1	2	10	5
5 ←	P_2	5	15	3
4 ←	P_3	10	30	3.33
3 ←	P_4	6	20	3.33
2 ←	P_5	4	18	4.55

Capacity = 10 kg

Sitagil M®

#



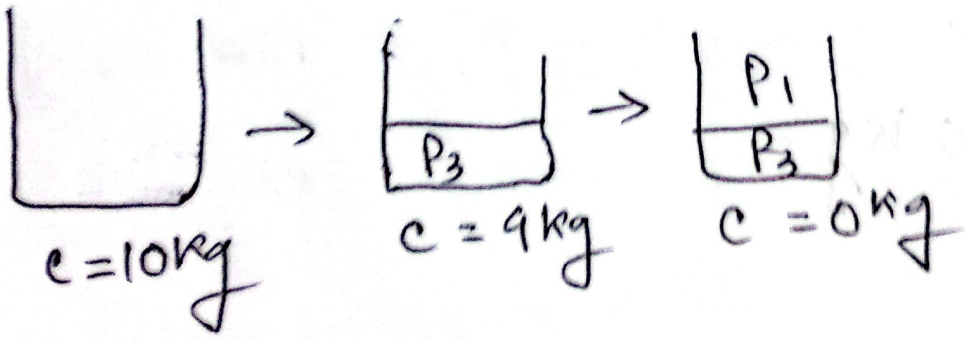
w: 2+4 = 6kg

v: 10 + 18 = 28 $

#

Product	weight	value	value/weight
3← P_1	4	20	5
2← P2	5	30	6
1← P_3	6	40	6.67

capacity = 10 kg



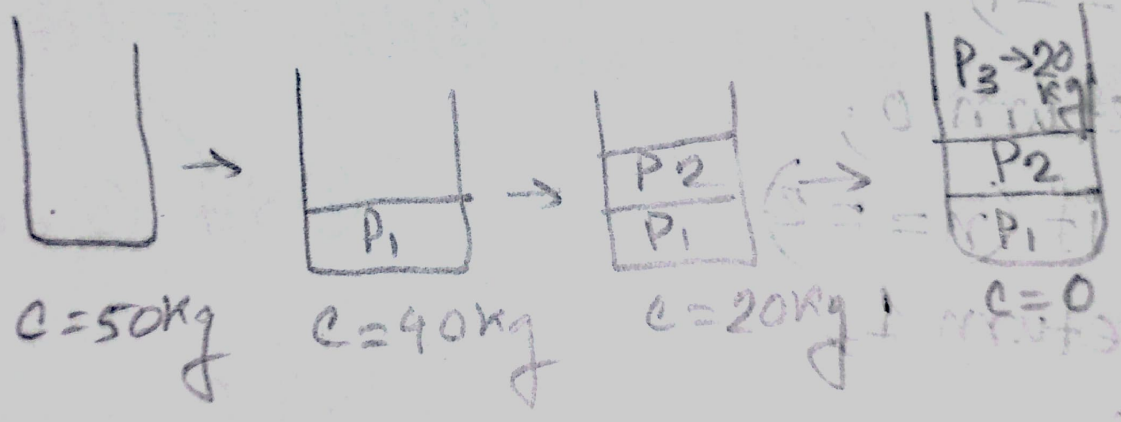
∴ w: 6+4=10kg

v: 40+20=60$

fractional knapsack

Product	weight	value	v/w
$1 \leftarrow P_1$	10	60	6
$2 \leftarrow P_2$	20	100	5
$3 \leftarrow P_3$	30	120	4

capacity = 50 kg.



$\therefore$ w: 10 + 20 + 20 = 50kg

v: 60 + 100 + 80 = 240 $

*Greedy never fails in fractional knapsack.

Sitagil M

Dynamic Programming

fibonacchi series:

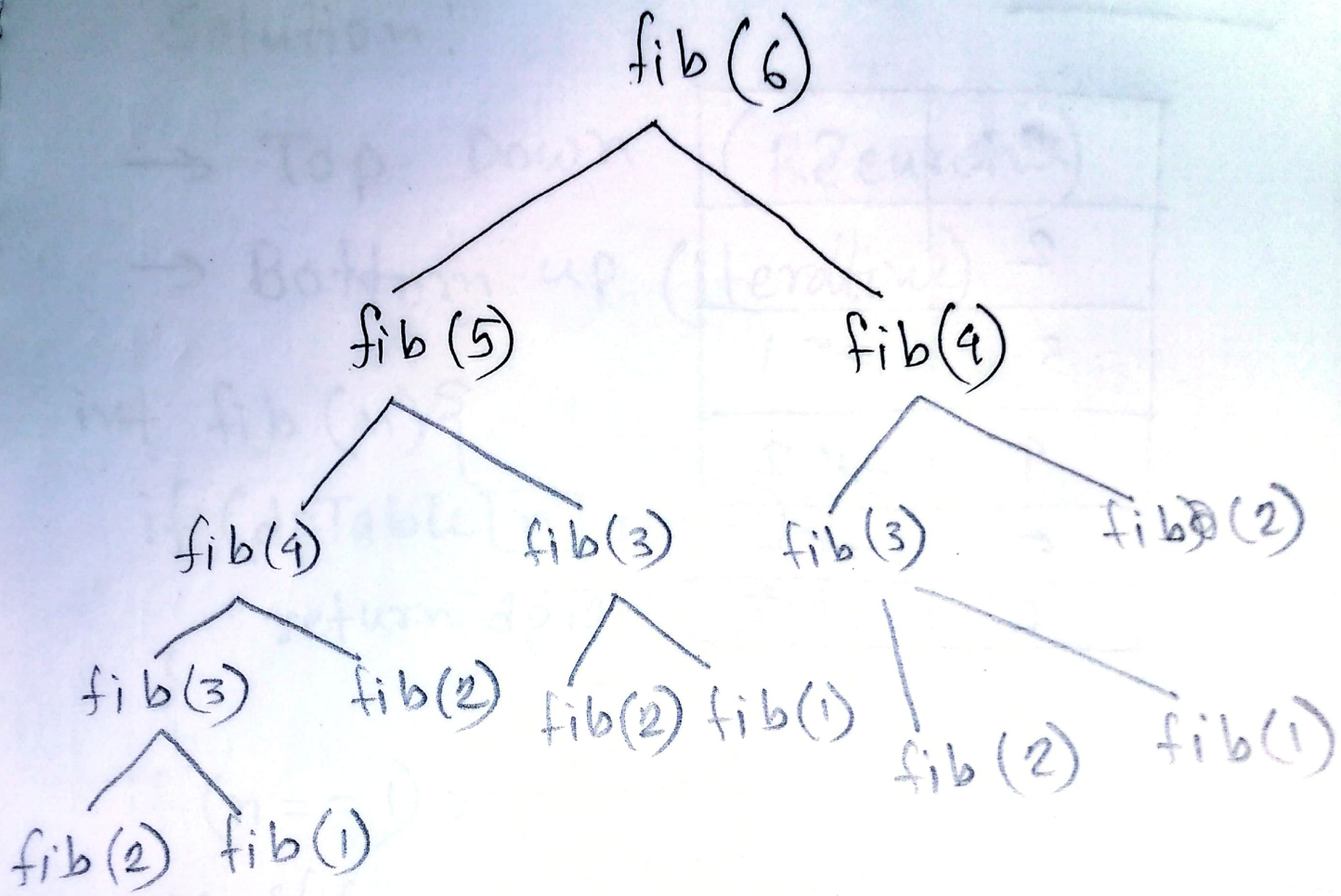
0 1 1 2 3 5 8 13 21

1st 2nd

3 4 _ _ _ _

```
int fib (int n) {
  if (n==1)
     return 0;
  else if (n==2)
     return 1;
  else{
     int fnd1 = fib (n-1);
     int fnd2 = fib (n-2);
     int result = fnd1 + fnd2;
     return result;
  }
}
```

if $n = 6$.



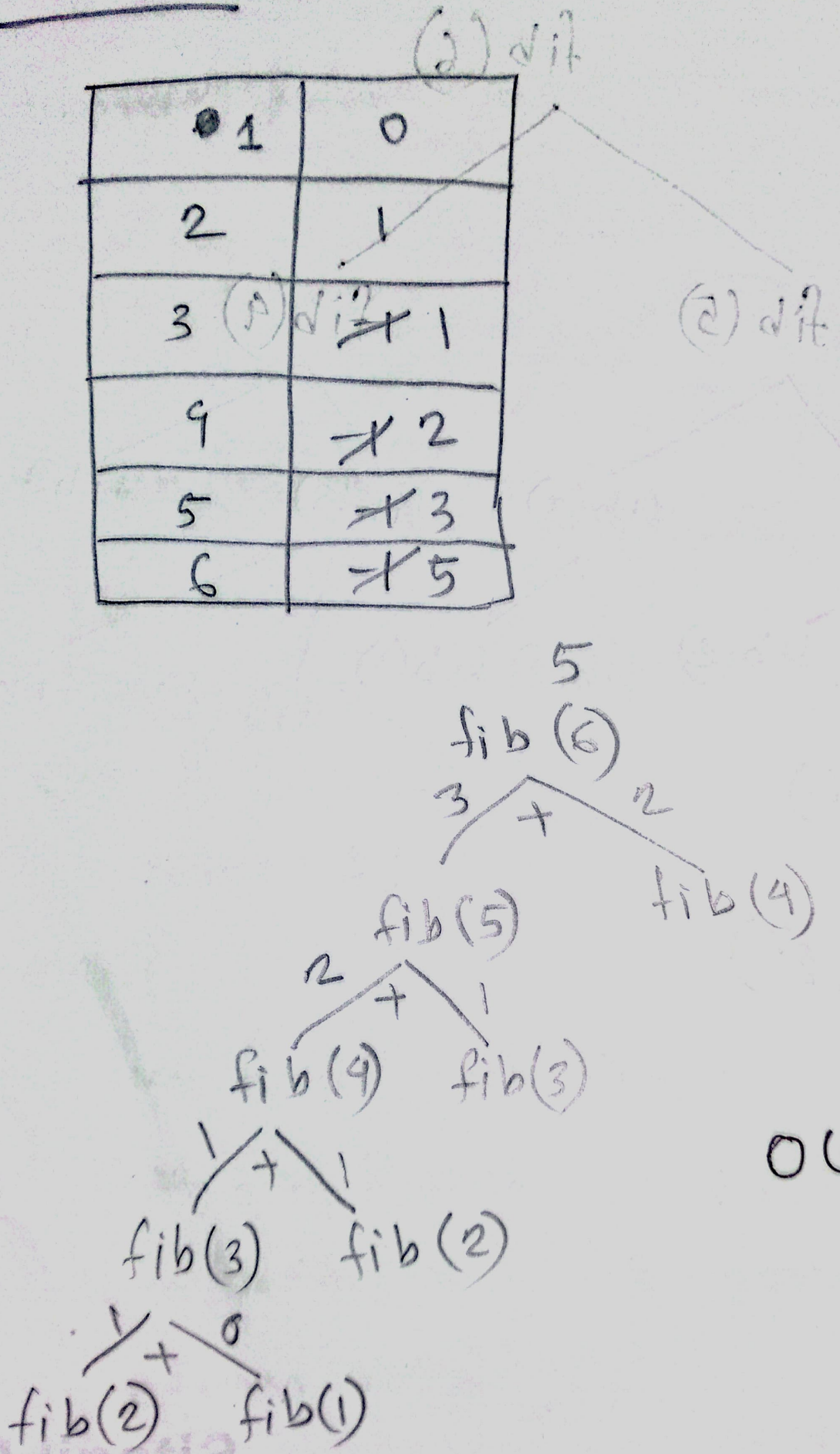
$O(2^n)$

Sitagil M®

DP Table

Top Down (Recursive)

1	0
2	1
3	~~1~~ 1
4	~~1~~ 2
5	~~1~~ 3
6	~~1~~ 5



$O(n)$

2 ways to implement a DP

Solution:

→ Top Down (Recursive)

→ Bottom up (iterative)

```
int fib(n){
  if (dpTable[n] != -1){
      return dpTable[n];
  }
  if (n == 1) return 0;
  else if(n == 2) return 1;
  else{
     int fnd1 = fib(n-1)
     int fnd2 = fib(n-2)
     int result = fnd1+fnd2
     dpTable[n] = result;
     return result;
  }
}
```

Sitagil M®

DP Applys for overlaping sub-problem.

DSA II - L-11 - 30 oct

Dynamic Programming

coin Chang.

optimal substructural property

ammount = 11

coins = [1, 5, 6, 8]

coin \ amount	0	1	2	3	4	5	6	7	8	9	10	11
1	0	1	2	3	4	5	6	7	8	9	10	11
5	0	1	2	3	4	1	2	3	4	5	2	3
6	0	1	2	3	4	1	1	2	3	4	2	2
8	0	1	2	3	4	1	1	2	1	2	2	2

whether we'll take it or not:

amount = 5 - 5 = 0

take: 1 + 0 = 1

not take: 5

Sitagil M®

$\boxed{x = 0 \times 7f7f}$

amount = 6

coins = [2, 3, 5]

coins \ amount	0	1	2	3	4	5	6
2	0	x	1	x	2	x	3
3	0	x	1	1	2	2	2
5	0	x	1	1	2	1	2