DAAPY 7

Saturday, February 25, 2023 7:22 PM

HW 5

Sunday, February 26, 2023 9:01 AM

9:02 → 9:15 ← 100

Solution

Don't Submit 90

100

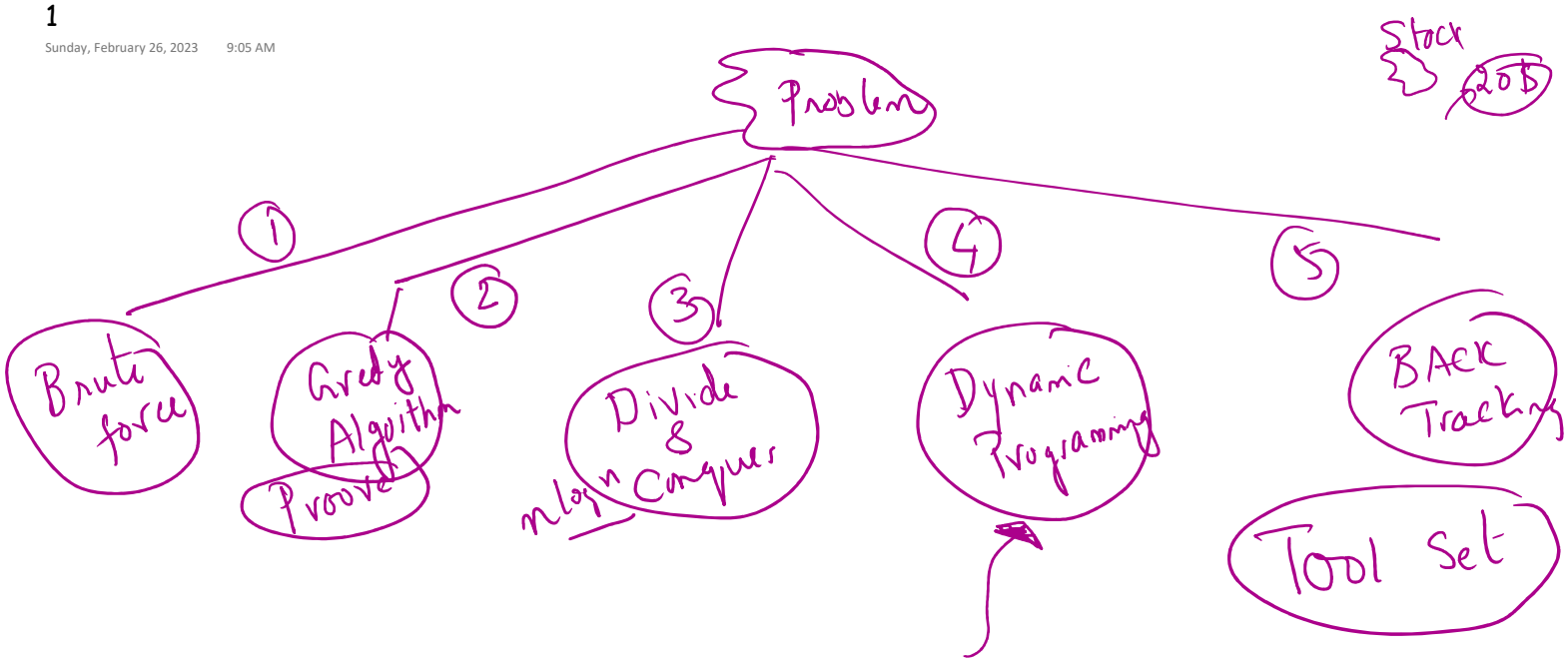
hw5 30 ~~50~~ 30

Stack 50

100

1

Sunday, February 26, 2023 9:05 AM



2

Sunday, February 26, 2023 9:10 AM

23 cents

100 → 77

23

77

1¢

n

12345

n

K

1 | 5 | 10 | ~~25~~

77

75

2

Minimal Greedy

∞ Coins

25 | 25 | 25 | 1 | 1

5 Coins

n = 5

0	1	2	3	4
25	25	25	1	1

ans

len(ans) = 5

n = 12345

Greedy Alg

~~1~~ | 5 | 10 | 25

13

10

96 cent

4 cent

23.919

23.92

3

Sunday, February 26, 2023 9:17 AM

K sol-

12 10 5 1

$n = 16$

Greedy Alg

16
12
4

12 1 1 1 1

5 Coin

Greedy Alg
① DESIGN
Difficult to Prove

10 5 1

4

Sunday, February 26, 2023 9:20 AM

Minimal

Coin

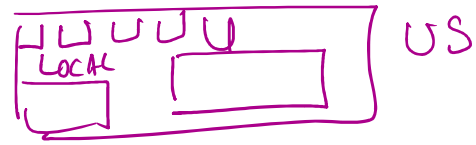
```
*/
public int calcMinNumStampsToFillRequest(int request)
{
    return 0;
}
}
```

7 stamps : K

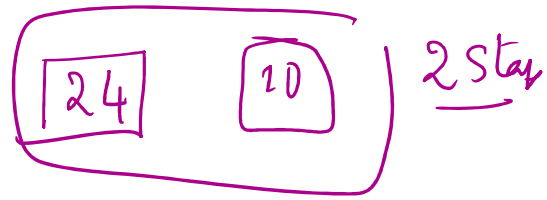
As an example, suppose an instance of `StampDispenser` was created with `stampDenominations`, {90, 30, 24, 10, 6, 2, 1}, and `calcMinNumStampsToFillRequest(int)` was called with request, 34. The call should return 2, as 34 cents can best be filled by one 24 cent stamp and one 10 cent stamp.

Things to keep in mind:

1. Assume that a junior programmer is going to read your code. You should include comments and any other aides that you use to communicate your code to other developers.
2. Optimize the code for speed.
3. The code should compile and work.
4. The code should work for countries with high denomination values where stamp values of 1000 or 9000 are common.



11567 stamps



5

Sunday, February 26, 2023 9:25 AM

```
*/
public int calcMinNumStampsToFillRequest(int request)
{
    return 0;
}
```

As an example, suppose an instance of `StampDispenser` was created with `stampDenominations` {90, 30, 24, 10, 6, 2, 1}, and `calcMinNumStampsToFillRequest(int)` was called with request, 34. The call should return 2, as 34 cents can best be filled by one 24 cent stamp and one 10 cent stamp.

Things to keep in mind:

1. Assume that a junior programmer is going to read your code. You should include comments and any other aides that you use to communicate your code to other developers.
2. Optimize the code for speed.
3. The code should compile and work.
4. The code should work for countries with high denomination values where stamp values of 1000 or 9000 are common.

6

Sunday, February 26, 2023 9:30 AM

Fibonacci number

F (25)

$$\boxed{\begin{aligned} F(0) &= 0 \\ F(1) &= 1 \end{aligned}}$$

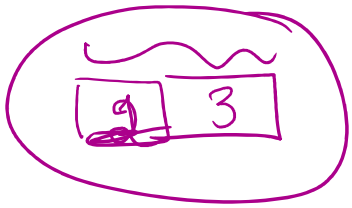
PrevPrev = 0

Prev = 1

$$F(n) = F(n-1) + F(n-2)$$

$$F(25) = \underline{F(24)} + F(23)$$

$$F(23+22) \quad (22+2$$



2	0	1	2	3	4	5
1	0	1	1	2	3	5

F(n) F(25) F(0) = 1
F(1) = 1

I BASE CASE
~~OPTIMAL~~

II Memoization
Memoization

Memoization

III OPTIMAL SUBSTRUCTURE

LIST	0	1	2	3	4	5	23	24	25
Array	0	1	1	2	3	5			?

UP TO 25

F(2) + F(1)

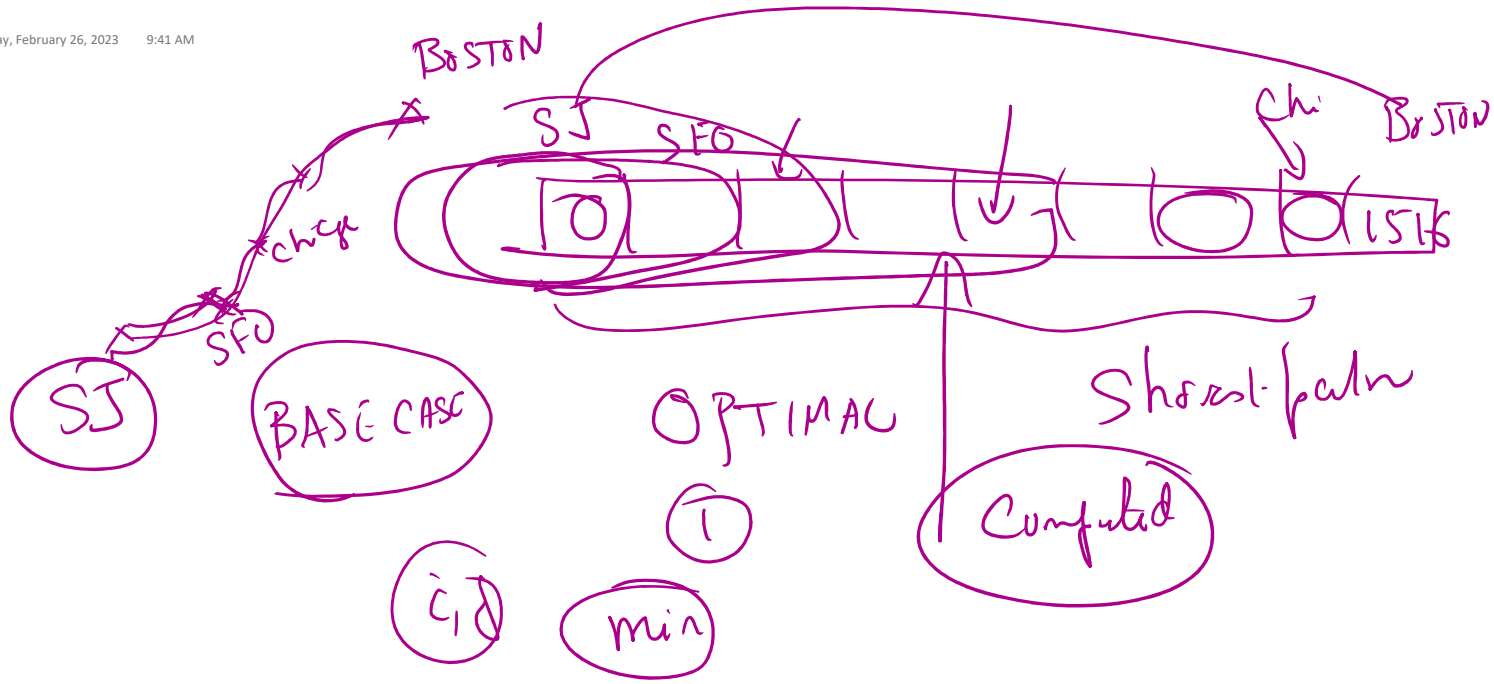
$\boxed{\text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2)}$

F(0)
F(1)

$\Theta(1)$ $\Theta(1)$ a[i]

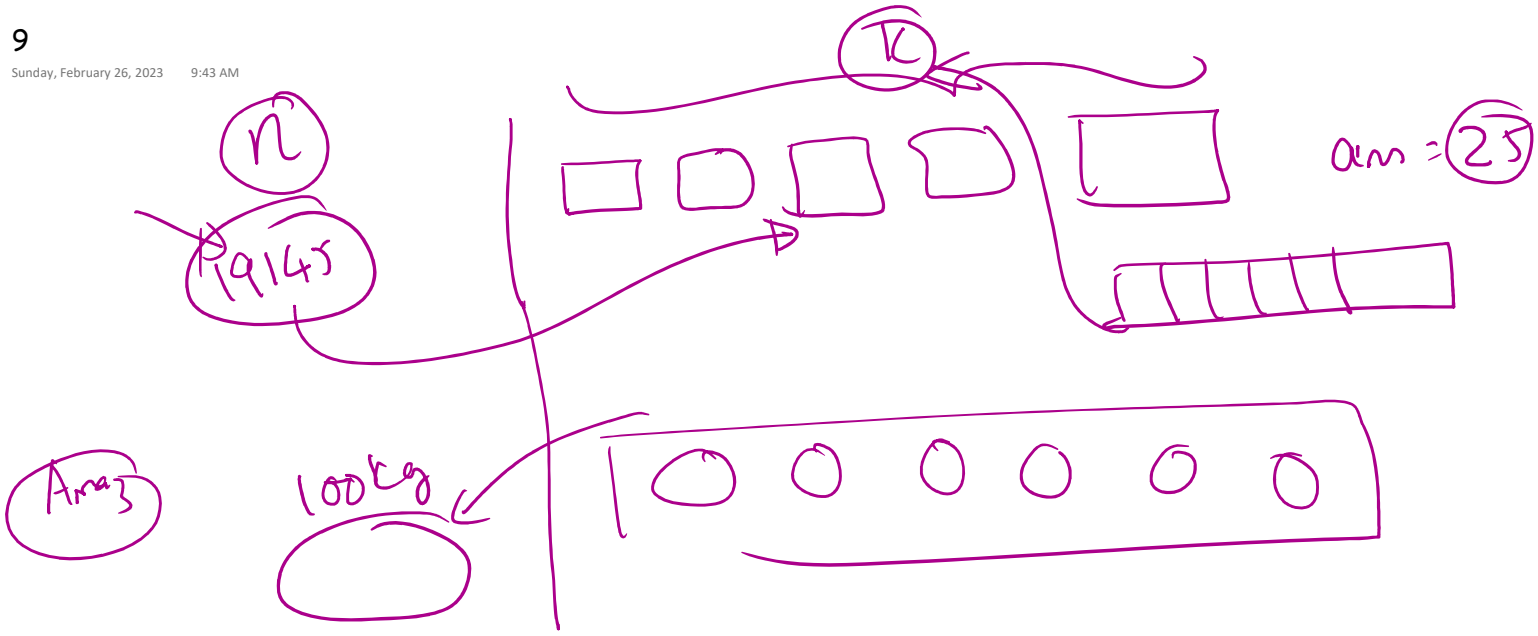
8

Sunday, February 26, 2023 9:41 AM



9

Sunday, February 26, 2023 9:43 AM



10

Sunday, February 26, 2023 9:45 AM

15123

6 | 1 3 4

You have infinite nu

2

3 3

BASE CASE

MEMOISZA Memoization

OPTIMAL SUB STRUCT

9

	0	1	2	3	4	5	6
Number of coin	0	1	2	1	1	2	2
K BASE	0	1	1	3	4	1	3

$O(n*n)$

$\Theta(n)$ SPACE

99 cents

1 Cent

SPACE $O(n)$

Time:

6

4 + 2

3 + 3

1 + 5

THIS MUST BE OPTIMAL & AVAIBLE $O(1)$

3

2

3

11

Sunday, February 26, 2023 10:02 AM

10

Sunday, February 26, 2023 9:45 AM

You have infinite nu

BASE CASE

MEMOISZA Memoization

OPTIMAL SUB STRUCT

ANS

	0	1	2	3	4	5	6
Number of coin	0	1	2	1	1	2	2
K BASE	0	1	1	3	4	1	3

99 cents

1 Cent

6

4 + 2

3 + 3

1 + 5

THIS MUST BE OPTIMAL & AVAILBLE

3

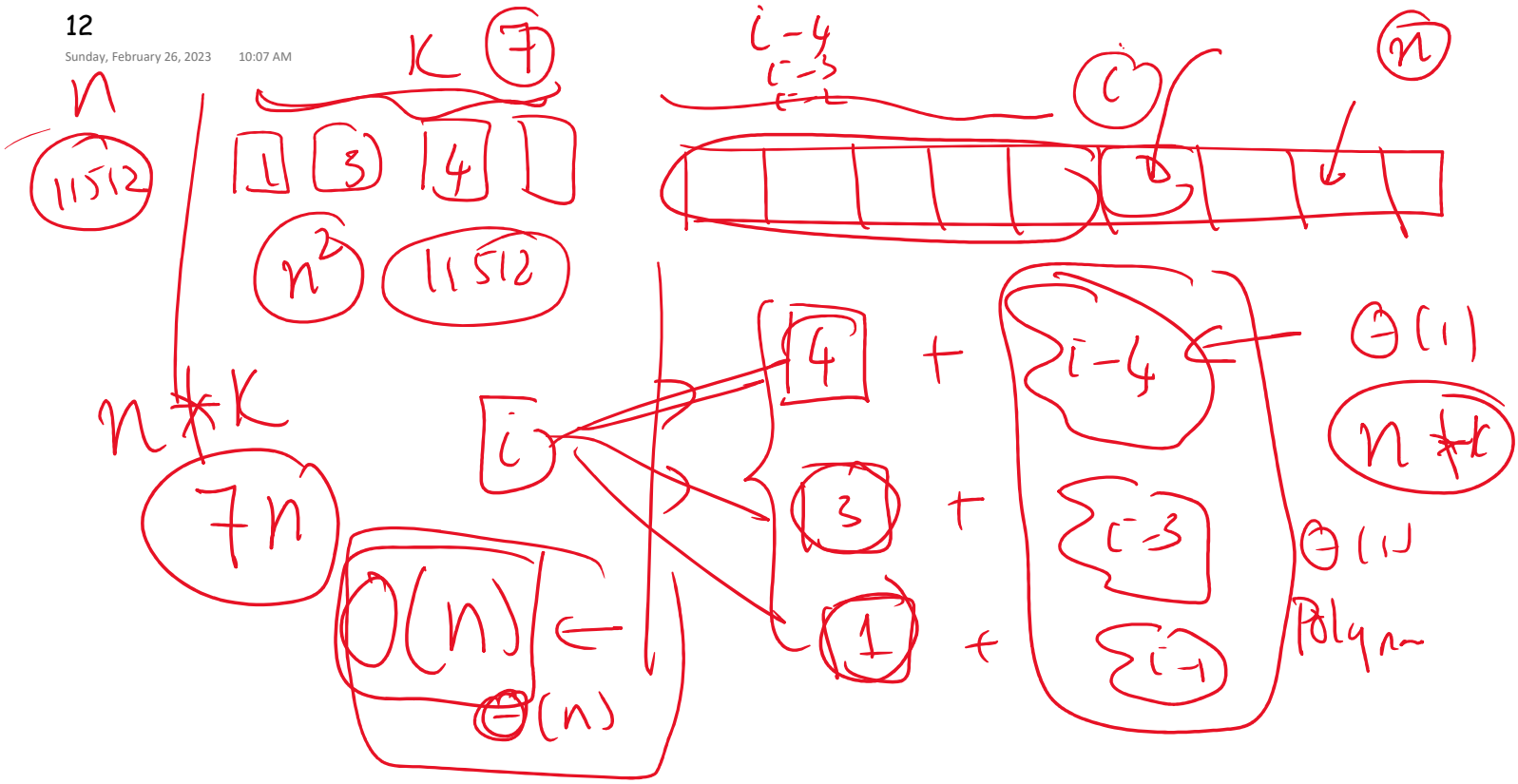
2

3

$\Theta(n)$ SPACE

12

Sunday, February 26, 2023 10:07 AM



13

Sunday, February 26, 2023 10:10 AM

n = 100 UPTO 100

```
Give minimum change for 34 cents using coins {1,2,6,10,24,30,90}
i =        0   1   2   3   4   5   6   7   8   9  10  11  12  13
m array    0   1   1   2   2   3   1   2   2   3   1   2   2   3
k array    0   1   2   1   2   1   6   1   2   1  10   1   2   1

          14  15  16  17  18  19  20
           3   4   2   3   3   4   2
           2   1   6   1   2   1  10

          20  21  22  23  24  25  26  27  28  29  30  31  32
           2   3   3   4   1   2   2   3   3   4   1   2   2
          10   1   2   1  24   1   2   1   2   1  30   1   2

minimum change for 23 cents can be achieved using 4 coin
1::Pick coin 1. Current val= 1. Remaining val= 22
2::Pick coin 2. Current val= 3. Remaining val= 20
3::Pick coin 10. Current val= 13. Remaining val= 10
4::Pick coin 10. Current val= 23. Remaining val= 0
```

Figure 17.4: Solution to Amazon Interview Question

Hop

29 Cents

1 2 2 24

29
1
28
2
26
2
24

1 2 6 10

19
1
18
2
16

16
6
10

19

14

Sunday, February 26, 2023 10:15 AM

```python
class Solution:
    ## YOU CANNOT CHANGE THIS INTERFACE
    ## LEETCODE INTERFACE
    def coinChange(self, coins: List[int], amount: int) -> int:
        ## YOU CANNOT CHANGE ANYTHING IN THIS PROCEDURE
        work = [0]
        changes = [] #If change cannot be given, you must put -1 in changes[0]
        show = False
        p = L0322(coins,amount,changes,work,show)
        num_change = len(changes)
        if (num_change == 1):
            if (changes[0] == -1):
                num_change = -1
        return num_change
```

{1,2,5}

15 16

O(n)

work = 0

| 5 | 5 | 5 | 2 | 2 | 2 |

0

work[0] = work[0] + 1

PASS BY VALU

V

K

n = 15 195

n = 8

| 2 | 5 | 10 | 2 |

15

Sunday, February 26, 2023 10:19 AM

(1, 2, 7) 1000 True False

Dynamic Pro

```python
class L0322:
    def __init__(self, coins: List[int], amount:'int', changes:'list of int', work:'List of size 1', show:'boolean'):
        self._d = coins
        self._n = amount
        self._ans = changes
        self._work = work
        self._show = show
        # YOU MUST GENERATE V table and k table
        self._v = []
        self._k = []
        # You can have any number of data structures here
        # MUST ERITE TWO ROUTINES
        self._alg()
        self._get_solution()

    def _increment_work(self):
        self._work[0] = self._work[0] + 1
```

16

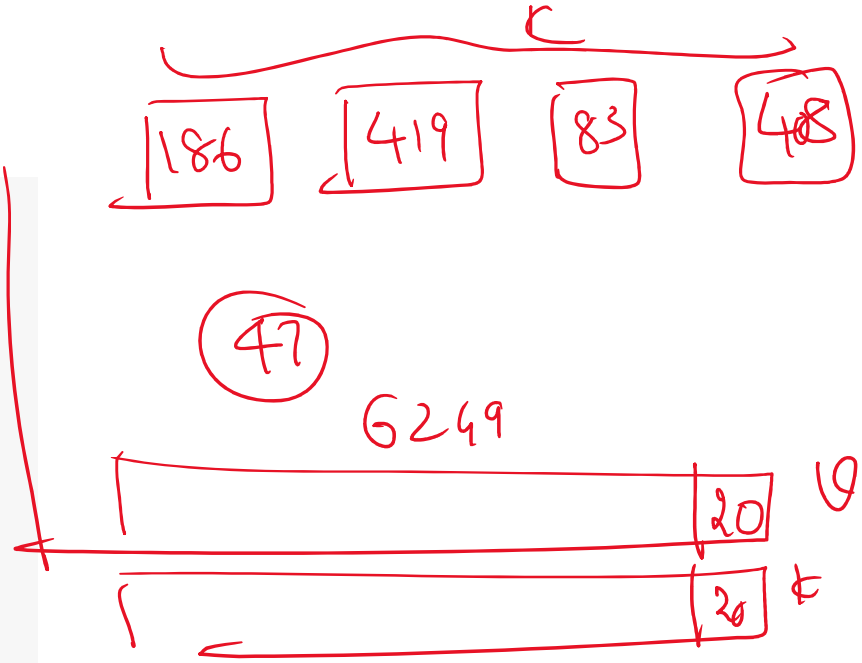
Sunday, February 26, 2023 10:22 AM

```
self._test1(w,c,e)

self._show = False
w = [186,419,83,408]
c = 6249
e = 20
self._test1(w,c,e)

self._show = False
w = [474,83,404,3]
c = 264
e = 8
self._test1(w,c,e)

self._show = False
w = [1]
c = 10000
e = 10000
self._test1(w,c,e)
```

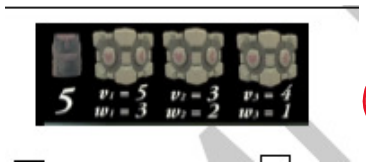


17

Sunday, February 26, 2023 10:26 AM

17.6 0/1 Knapsack problem

BACK PACK



5kg

	1	2	3
Value	5	3	4
WEIGHT	3	2	1
	TV	VCR	LAPTOP

3 items

9$

4kg

TV
LAPTOP

1 3 5 ∞

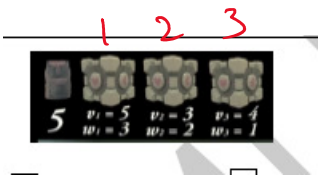
Steal or No steal

5 TV LAPT 9000$ 4kg ∞

8K 5kg

18

Sunday, February 26, 2023 10:53 AM



V

NO Item	0	1	2	3	4	5
	0	0	0	0	0	0
1	0	0	0	5	5	5
2	0	0	3	5	5	8
3	0	4	4	7	8	9

ITEM

BAGS'18

K

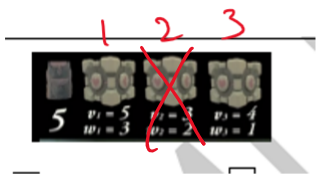
NO	0	1	2	3	4	5
	0	0	0	0	0	0
1	0	0	0	1	1	1
	0	0	1	0	0	1
	0	1	1	1	1	1

19

Sunday, February 26, 2023 11:15 AM

18

Sunday, February 26, 2023 10:53 AM



K	0	1	2	3	4	5
NO	0	0	0	0	0	0
1	0	0	0	1	1	1
2	0	0	1	0	0	1
3	0	1	1	1	1	1

ITEM

V	0	1	2	3	4	5
NO Item	0	0	0	0	0	0
1	0	0	0	5	5	5
2	0	0	3	5	5	8
3	0	4	4	7	8	9

BAGS'18

CLR

9$

$ 4 5
W 1 3

9$

5kg

4kg

1kg

1kg

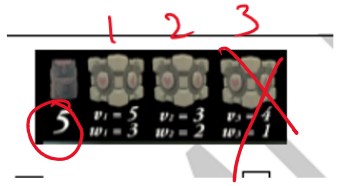
i6

20

Sunday, February 26, 2023 11:20 AM

18

Sunday, February 26, 2023 10:53 AM



V

NO Item	0	1	2	3	4	5
	0	0	0	0	0	0
1	0	0	0	5	5	5
2	0	0	3	5	5	8
3	0	4	4	7	8	9

BAG SIZE

ITEM

	1	2
$	4	5
W	1	3

K

NO	0	1	2	3	4	5
	0	0	0	0	0	0
1	0	0	0	1	1	1
2	0	0	1	0	0	1
3	0	1	1	1	1	1

9 $

3

9 $

4 kg

~~4 kg~~ 1 kg

5 kg

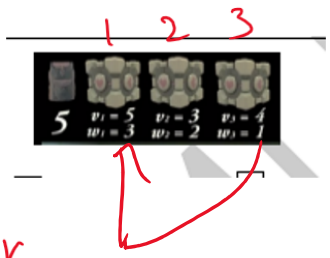
1 kg

21

Sunday, February 26, 2023 11:22 AM

18

Sunday, February 26, 2023 10:53 AM



V

ITEM	0	1	2	3	4	5
	0	0	0	0	0	0
	0	0	0	5	5	5
	0	0	3	5	5	8
	0	4	4	7	8	9

BAG SIZE

NO Item

1 $\frac{5}{3}$

2 $\frac{3}{2}$

3 $\frac{4}{1}$

K

NO	0	1	2	3	4	5
	0	0	0	0	0	0
1	0	0	0	1	1	1
	0	0	1	0	0	1
	0	1	1	1	1	1

1 3

2 1

3 2

OPTIMAL

1

3

2

1, 2, 3
1, 3, 2
2, 1, 3
2, 3, 1
3, 1, 2
3, 2, 1

NO

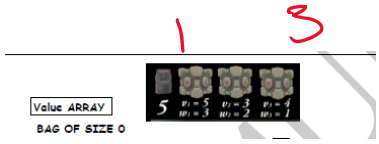
2

1

1

22

Sunday, February 26, 2023 11:24 AM



Item

5
3

4
1

No item

1

2

3

6 Pages

1	2	3	1
1	3	2	2
2	1	3	3
2	3	1	4
3	1	2	5
3	2	1	6

9 $

23

Sunday, February 26, 2023 11:26 AM

BAG

No item \	0	1	2	3	4	5
	0	2	3	4	5	6
2	7	8	9	$\boxed{10}$	11	12
1	13	14	15	16	17	18
3	19	20	21	22	23	24

Wt

1
2
3
4
21

9

0

6 PAGES

1: 0 (No item in kwpl. bag 5x)

$\boxed{10}$:

24

Sunday, February 26, 2023 11:31 AM

30 HAR

	0	1	2	3	4	5
a	5	1	0	4	2	3

```
int H (a, n) {
    if (a[n] = = n)
        return 0
    int t = a[n]
    a[n] = a[t]
    int l = 1 + H(a, n)
    a[n] = t
    return l;
}
```

K

n!

3

100

50

a = {5, 1, 0, 4, 2, 3}

n = 3

l = H(a, n)

l = 4

l = 4

3

0	1	2	3	4	5
5	1	0	4	2	3

2

0

5

~~3~~

3

0

2

4

a

n=3

a

n=3

t = 3

a[3] = 3

l = 1 + H(a, 3)

a[3] = 3 return l

1

0

a

n=3

t = 0

a[3] = a[0]

l = 1 + H(a, 3)

a[3] = 0 return l

2

a

n=3

t = 2

a[3] = a[2]

l = 1 + H(a, 3)

a[3] = 3

return l

a[ ]

n=3

t = 4

a[3] = a[4]

l = 1 + H(a, 3)

a[3] = 4

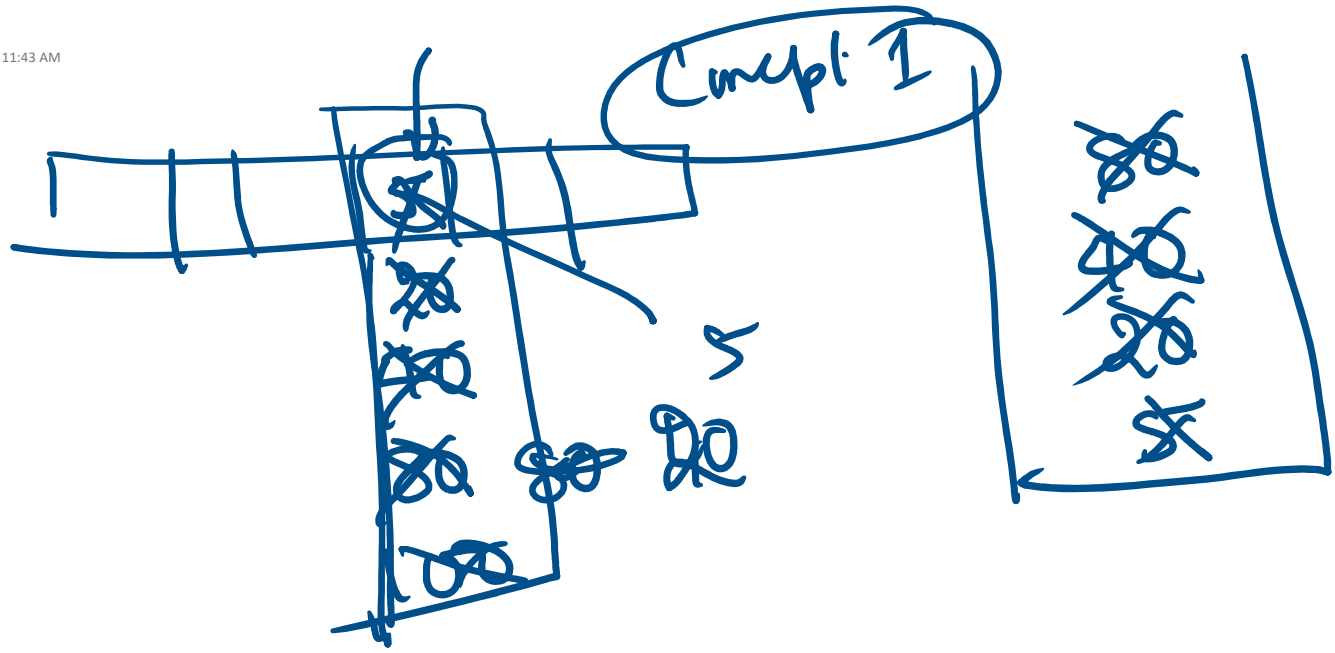
return l

l=4

4

25

Sunday, February 26, 2023 11:43 AM



26

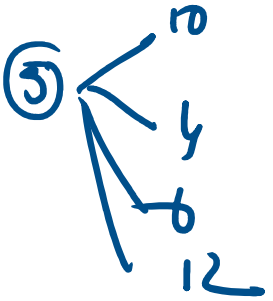
Sunday, February 26, 2023 11:45 AM

```
#########################################
# TIME:THETA(N^2)
# Space:THETA(1)
# I have implemented the code.  NOTHING CAN BE CHANGED HERE
#########################################
def nsquare_time_constant_space(self, a: List[int]) -> "[sellday,buyday,work]":
    n = len(a)
    gp = 0
    bday = 0
    sday = 0
    work = 0
    for i in range(0, n - 1):
        for j in range(i + 1, n):
            work = work + 1
            p = a[j] - a[i]
            if p > gp:
                gp = p
                bday = i
                sday = j
    return [sday, bday, work]
```

n^2

$O(n^2)$

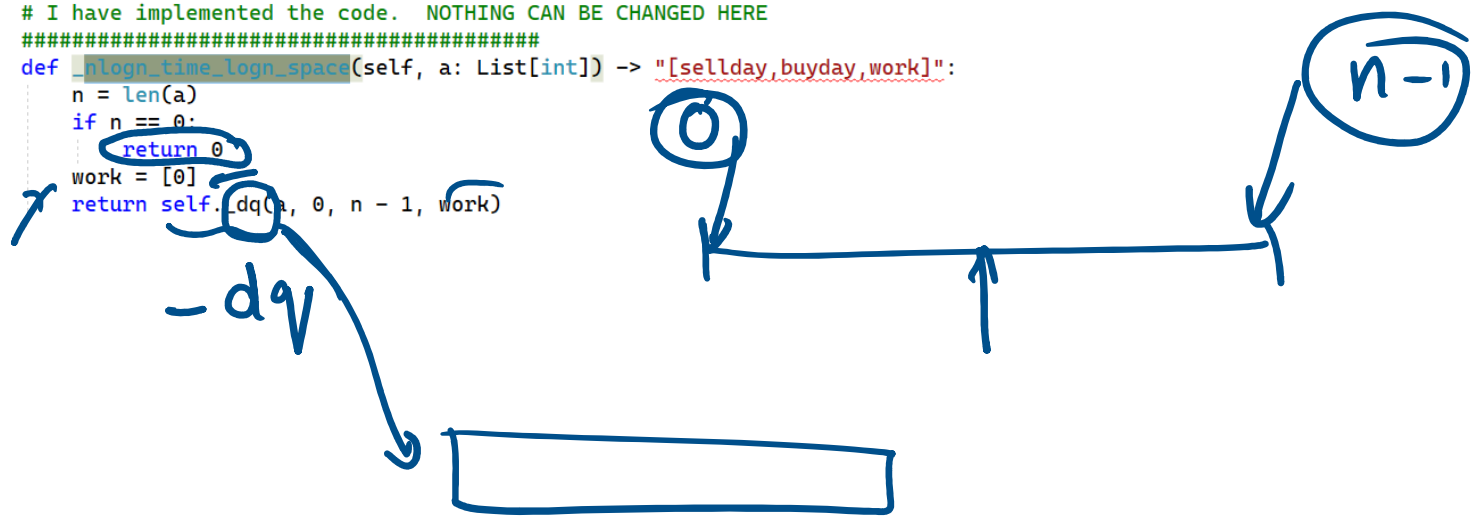
day	0	1	2	3	4
cost	5	10	4	6	12



27

Sunday, February 26, 2023 11:50 AM

```
#########################################
# TIME:THETA(NlogN)
# Space:THETA(logn)
# I have implemented the code.  NOTHING CAN BE CHANGED HERE
#########################################
def _nlogn_time_logn_space(self, a: List[int]) -> "[sellday,buyday,work]":
    n = len(a)
    if n == 0:
        return 0
    work = [0]
    return self._dq(a, 0, n - 1, work)
```



28

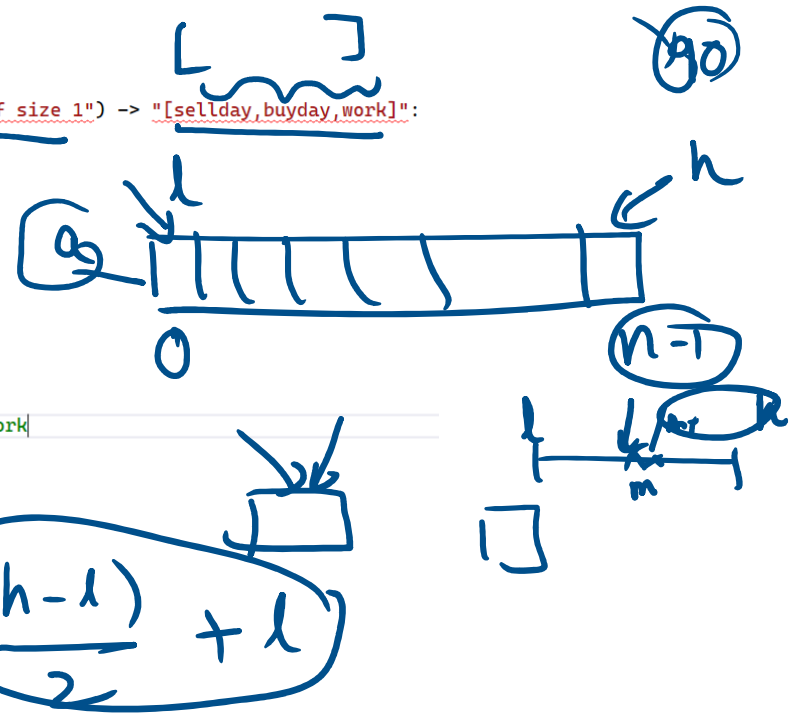
Sunday, February 26, 2023 11:53 AM

```python
def _dq(self, a: List[int], l: "int", h: "int", work: "list of size 1") -> "[sellday,buyday,work]":
    if l == h:
        # exactly one element ;
        # Profit is zero
        ans = [h, l, work[0]]
        return ans

    # Divide
    work[0] = work[0] + 1
    m = ((h - l) // 2) + l
    # Left side profit index
    li = self._dq(a, l, m, work)  # li =sellday,buyday,work
    # Right side profit index
    ri = self._dq(a, m + 1, h, work)  # ri = sellday,buyday,work
```

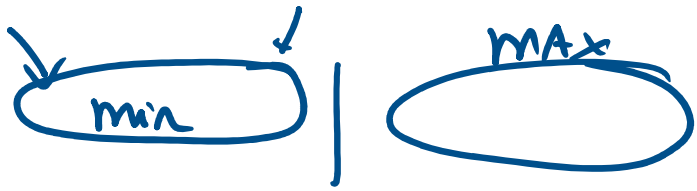
$$m = \left(\frac{h+l}{2}\right)$$

$$\frac{(h-l)}{2} + l$$



29

Sunday, February 26, 2023 11:58 AM




```python
# CONQUER
work[0] = work[0] + 1

# Find minimum number index on left side
minlefti = self._get_min_or_max_index(a, l, m, work, self._min)
# Find maximum number index on right side
maxrighti = self._get_min_or_max_index(a, m + 1, h, work, self._max)

crossprofit = self._compute_profit(a, maxrighti, minlefti)
leftprofit = self._compute_profit(a, li[0], li[1])
rightprofit = self._compute_profit(a, ri[0], ri[1])

# First figure out left or right is profitable
leftrightprofitindex = li  # sellday,buyday,work
leftrightprofit = leftprofit
if rightprofit > leftprofit:
    leftrightprofitindex = ri  # sellday,buyday,work
    leftrightprofit = rightprofit
```

30

Sunday, February 26, 2023 12:00 PM

```
##########################################
def _get_min_or_max_index(self, a: List[int], s: "int", m: "int", work: "list of size 1", minormax: "int") -> "int":
    v = a[s]
    vi = s
    for i in range(s + 1, m + 1):
        work[0] = work[0] + 1
        if minormax == self._min:
            if a[i] < v:
                v = a[i]
                vi = i
        else:
            if a[i] > v:
                v = a[i]
                vi = i
    return vi  # return index
```

31

Sunday, February 26, 2023 12:04 PM

max

0 1 2
[S, B, W]

```
crossprofit = self._compute_profit(a, maxrighti, minlefti)
leftprofit = self._compute_profit(a, li[0], li[1])
rightprofit = self._compute_profit(a, ri[0], ri[1])

# First figure out left or right is profitable
leftrightprofitindex = li  # sellday,buyday,work
leftrightprofit = leftprofit
if rightprofit > leftprofit:
    leftrightprofitindex = ri  # sellday,buyday,work
    leftrightprofit = rightprofit

## Now compare with cross profit
if crossprofit > leftrightprofit:
    ans = [maxrighti, minlefti, work[0]]
    return ans
else:
    ans = [leftrightprofitindex[0], leftrightprofitindex[1], work[0]]
    return ans
```

L
R

32

Sunday, February 26, 2023 12:07 PM

```
##########################################
def _ntime_constant_space(self, a: List[int]) -> "[sellday,buyday,work]":
    n = len(a)
    if n == 0:
        return [0, 0, 0]
    work = 1
    gp = 0
    sellday = 0
    buyday = 0
    lowest_stock_day = 0
    lowest_stock_day_price = a[0]
    for i in range(1, n):
        work = work + 1
        price_today = a[i]
        if price_today < lowest_stock_day_price:
            lowest_stock_day = i
            lowest_stock_day_price = price_today
        else:
            p = self._compute_profit(a, i, lowest_stock_day)
            if p > gp:
                gp = p
                buyday = lowest_stock_day
                sellday = i
    ans = [sellday, buyday, work]
    return ans
```

$\Theta(n)$