DAAPY 7

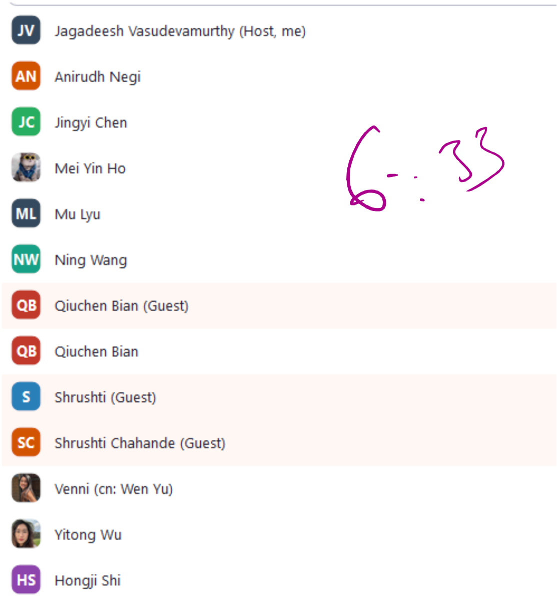
Wednesday, October 18, 2023 4:50 PM

One Student → Zoom 5PM +

Submit on next-wed

Zoom { → Sunday - 9PM PST →

6:50 Record



Jagadeesh Vasudevamurthy (Host, me)

Anirudh Negi

Jingyi Chen

Mei Yin Ho

Mu Lyu

Ning Wang

Qiuchen Bian (Guest)

Qiuchen Bian

Shrushti (Guest)

Shrushti Chahande (Guest)

Venni (cn: Wen Yu)

Yitong Wu

Hongji Shi

6:33

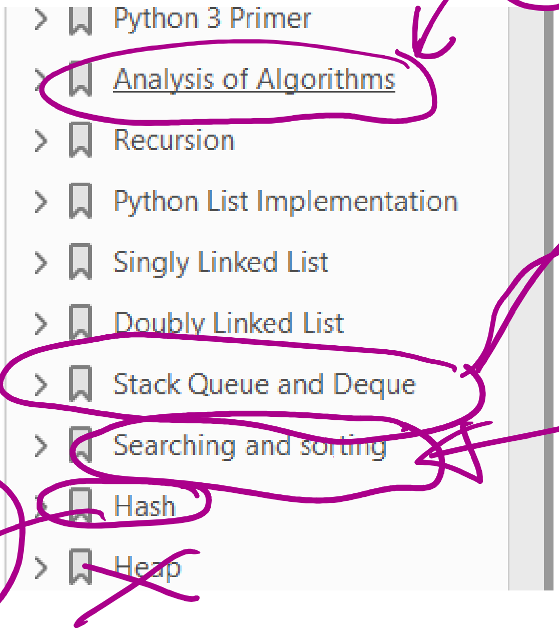
complexity

Wednesday, October 18, 2023 6:10 PM

	BUS Dynamic Array [ ]	TRAIN SLIST	STACK	Queue	Deque	HASh
Insert	append: $\Theta(1)$ AMortized Prepend $O(n)$ Anyplace $O(n)$	A $\Theta(1)$ P $\Theta(1)$ Anyplc $O(n)$	Push $\Theta(1)$	Enque $\Theta(1)$	Insert F Insert B $\Theta(1)$	$\Theta(1)$
a[i]	$\Theta(1)$	$O(n)$	X	X	X	X
FIND	$O(n)$	$O(n)$	TOP $\Theta(1)$	FRONT BACK $\Theta(1)$	FRONT BACK $\Theta(1)$	$\Theta(1)$
Delete	$O(n)$	$O(n)$	POP $\Theta(1)$	deque $\Theta(1)$	Delete F $\Theta(1)$ Delete B	$\Theta(1)$
Min	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$!!!	X
MAX	$\Theta(n)$	$\Theta(n)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	X

1

Wednesday, October 18, 2023 5:02 PM



- Python 3 Primer
- Analysis of Algorithms
- Recursion
- Python List Implementation
- Singly Linked List
- Doubly Linked List
- Stack Queue and Deque
- Searching and sorting
- Hash
- ~~Heap~~

5

Midterm

1 HW

2

4

3

Dynamic Programi

2

Wednesday, October 18, 2023 5:05 PM

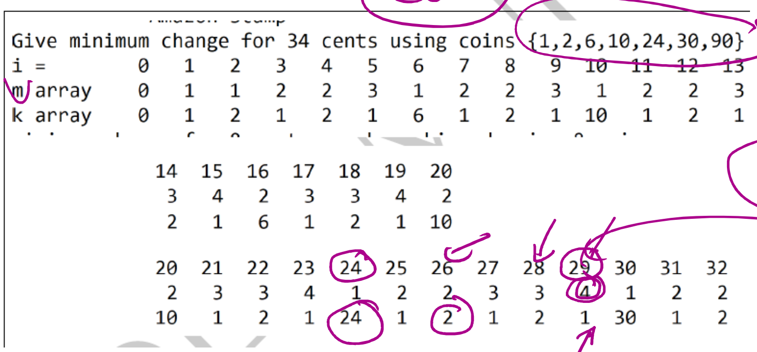
Coins

K

n = 195

n = 29

4



```
Amazon Stamp
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
```

1			

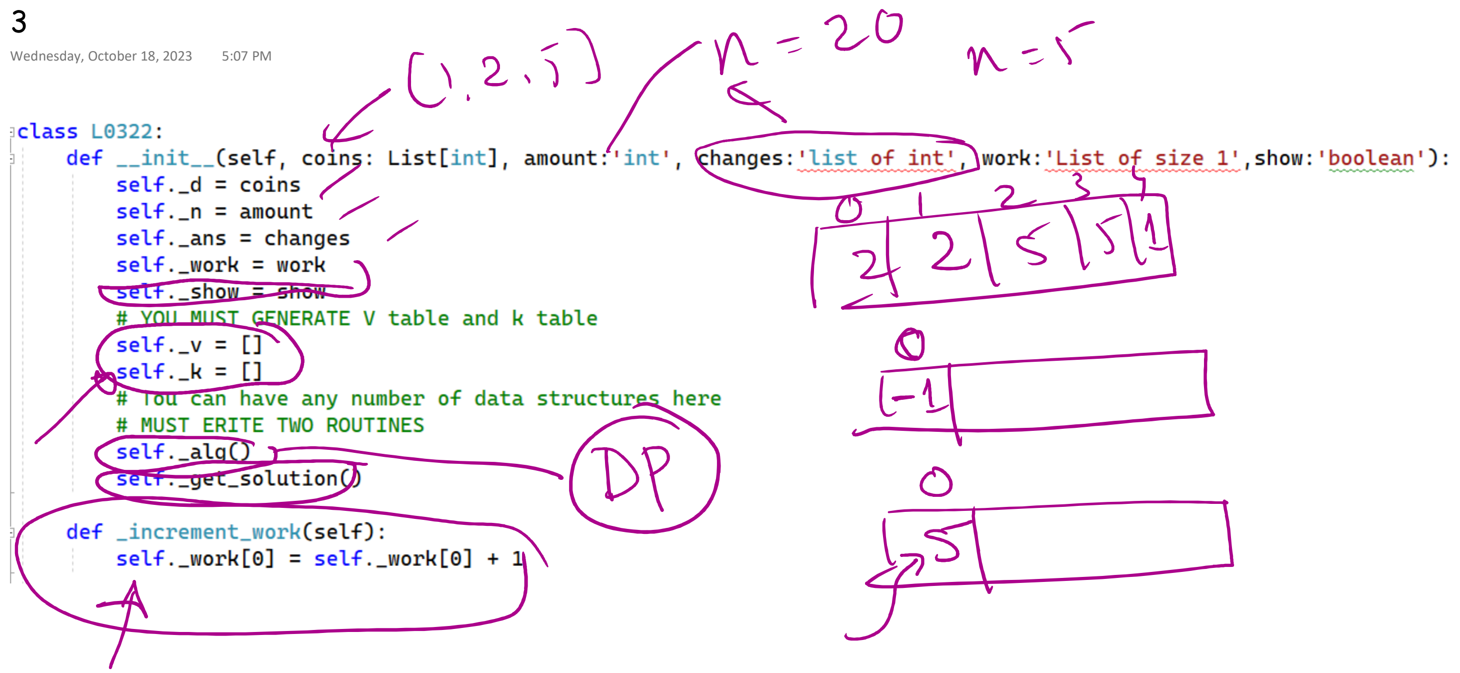
1 2 2 24

29

$$\begin{array}{r} 29 \\ -\ 3 \\ \hline 26 \\ -\ 2 \\ \hline 24 \end{array}$$

3

Wednesday, October 18, 2023 5:07 PM

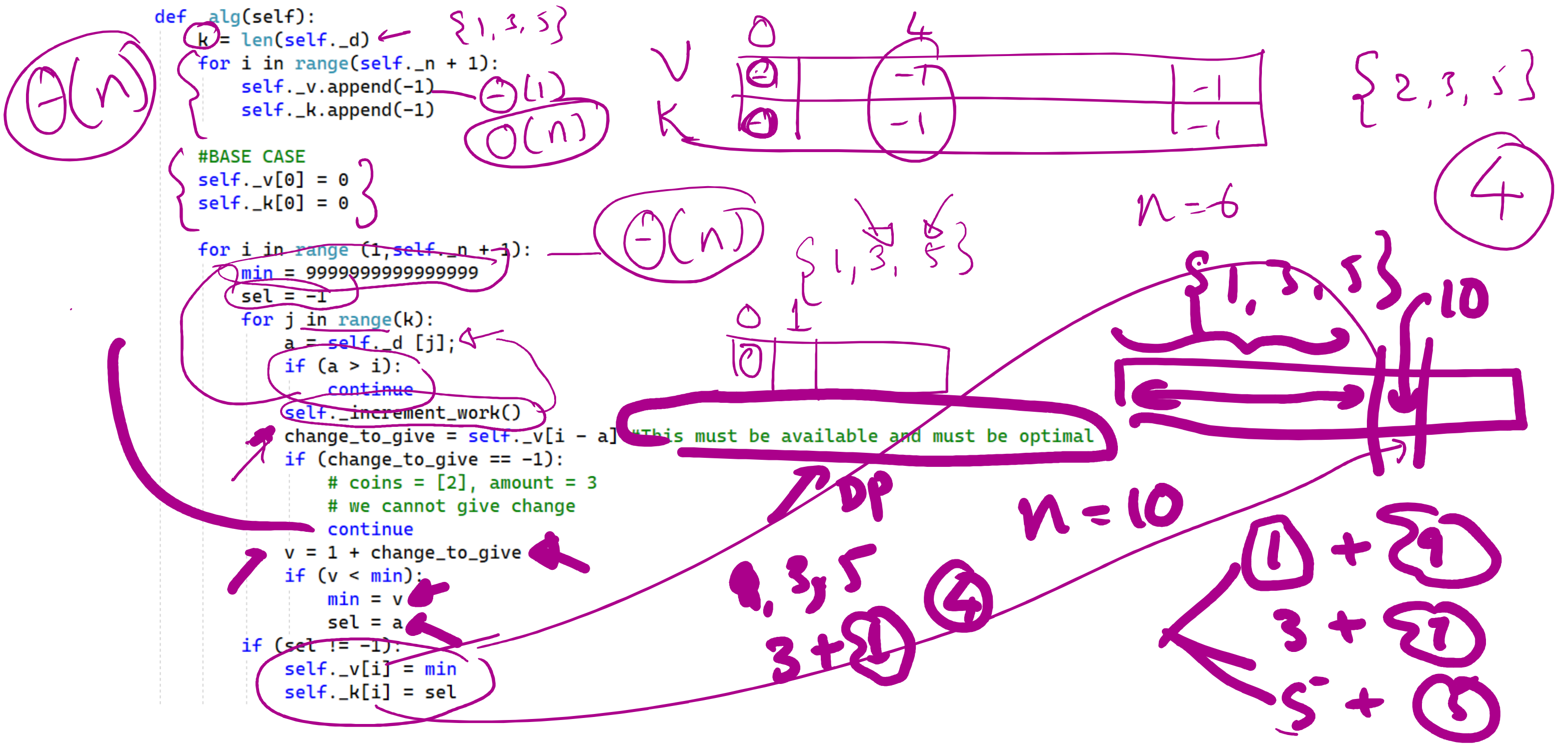


```
class L0322:
    def __init__(self, coins: List[int], amount:'int', changes:'list of int', work:'List of size 1',show:'boolean'):
        self._d = coins
        self._n = amount
        self._ans = changes
        self._work = work
        self._show = show
        # YOU MUST GENERATE V table and 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
```

4

Wednesday, October 18, 2023 5:09 PM



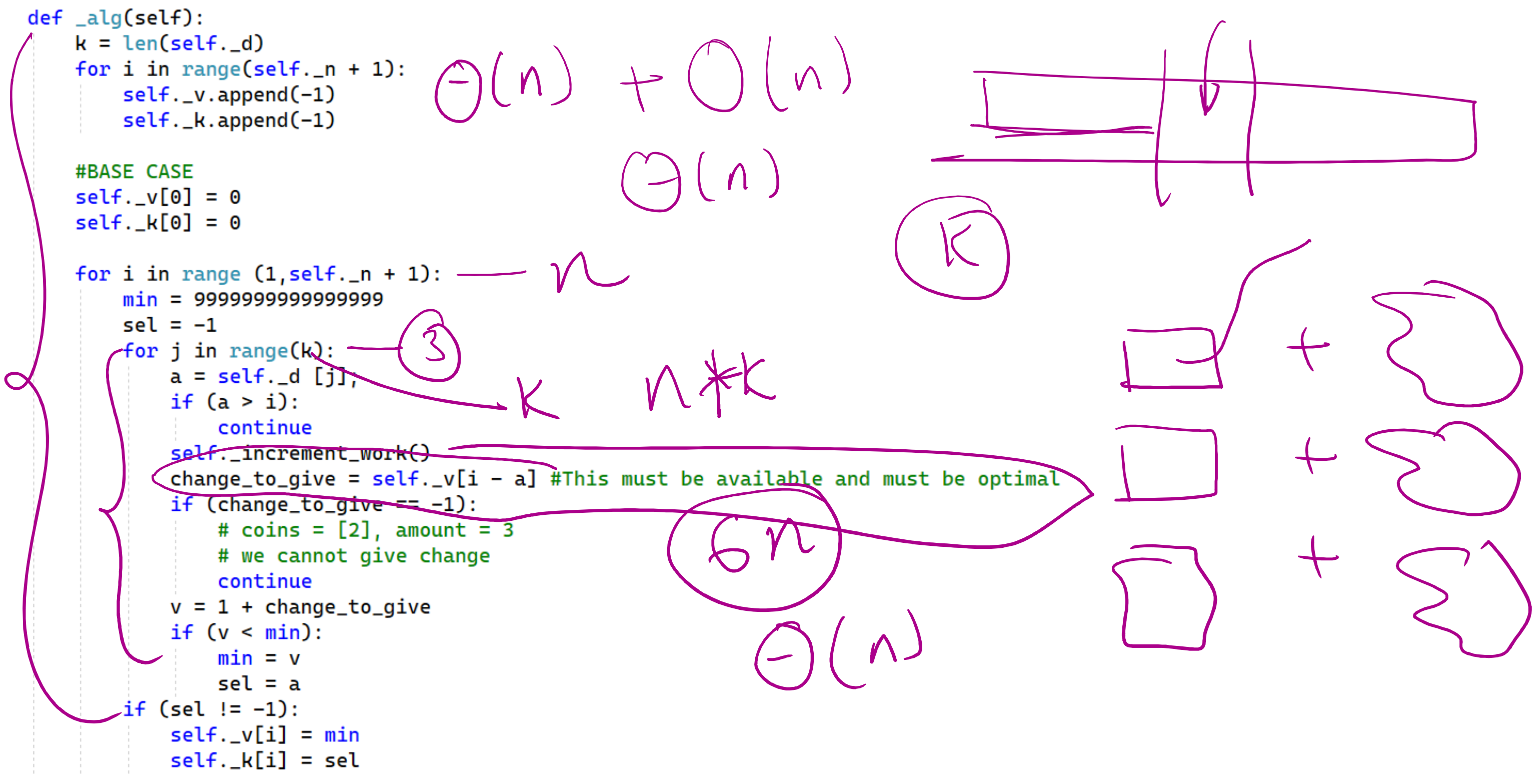
```
def _alg(self):
    k = len(self._d)
    for i in range(self._n + 1):
        self._v.append(-1)
        self._k.append(-1)

    #BASE CASE
    self._v[0] = 0
    self._k[0] = 0

    for i in range (1,self._n + 1):
        min = 9999999999999999
        sel = -1
        for j in range(k):
            a = self._d [j];
            if (a > i):
                continue
            self._increment_work()
            change_to_give = self._v[i - a] #This must be available and must be optimal
            if (change_to_give == -1):
                # coins = [2], amount = 3
                # we cannot give change
                continue
            v = 1 + change_to_give
            if (v < min):
                min = v
                sel = a
        if (sel != -1):
            self._v[i] = min
            self._k[i] = sel
```

5

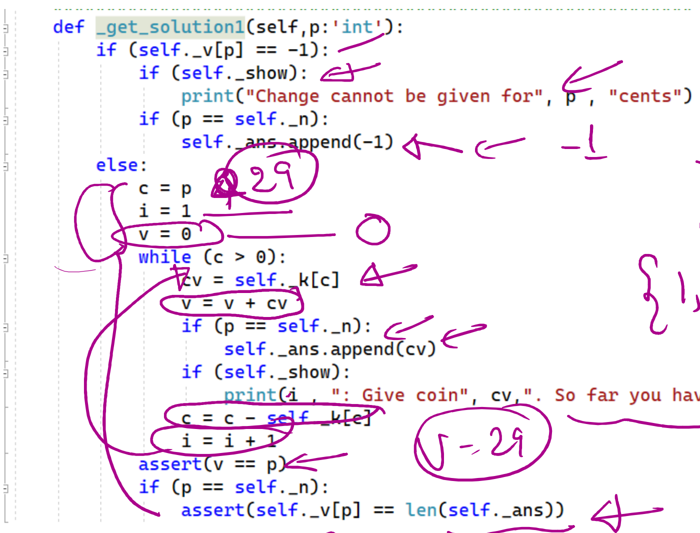
Wednesday, October 18, 2023 5:16 PM



```
def _alg(self):
    k = len(self._d)
    for i in range(self._n + 1):
        self._v.append(-1)
        self._k.append(-1)

    #BASE CASE
    self._v[0] = 0
    self._k[0] = 0

    for i in range (1,self._n + 1):
        min = 9999999999999999
        sel = -1
        for j in range(k):
            a = self._d [j];
            if (a > i):
                continue
            self._increment_work()
            change_to_give = self._v[i - a] #This must be available and must be optimal
            if (change_to_give == -1):
                # coins = [2], amount = 3
                # we cannot give change
                continue
            v = 1 + change_to_give
            if (v < min):
                min = v
                sel = a
        if (sel != -1):
            self._v[i] = min
            self._k[i] = sel
```



```
def _get_solution1(self,p:'int'):
    if (self._v[p] == -1):
        if (self._show):
            print("Change cannot be given for", p , "cents")
        if (p == self._n):
            self._ans.append(-1)
    else:
        c = p
        i = 1
        v = 0
        while (c > 0):
            cv = self._k[c]
            v = v + cv
            if (p == self._n):
                self._ans.append(cv)
            if (self._show):
                print(i , ": Give coin", cv,". So far you have given=",v,".Remaining to give",(c-cv)) ;
            c = c - self._k[c]
            i = i + 1
        assert(v == p)
        if (p == self._n):
            assert(self._v[p] == len(self._ans))
```

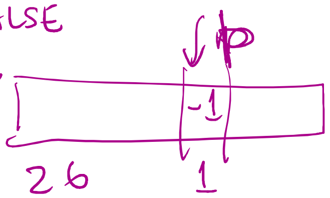
29

0

-1

V = 29

Show = FALSE
Show = True



-1

{1, 2, 26}

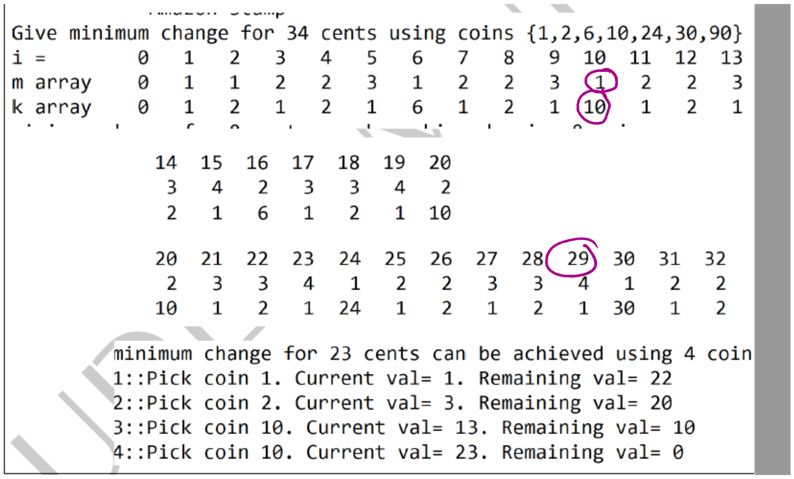
29 28 26
1 2 26
28 26 0

5

1 cents

1 cents

5

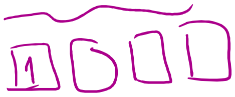


```
Give minimum change for 34 cents using coins {1,2,6,10,24,30,90}
i =         0   1   2   3   4   5   6   7   8   9  10  11  12  13
m array     0   1   1   2   2   3   1   2   2   3   1   2   2   3
k array     0   1   2   1   2   1   6   1   2   1  10   1   2   1

           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
```



7

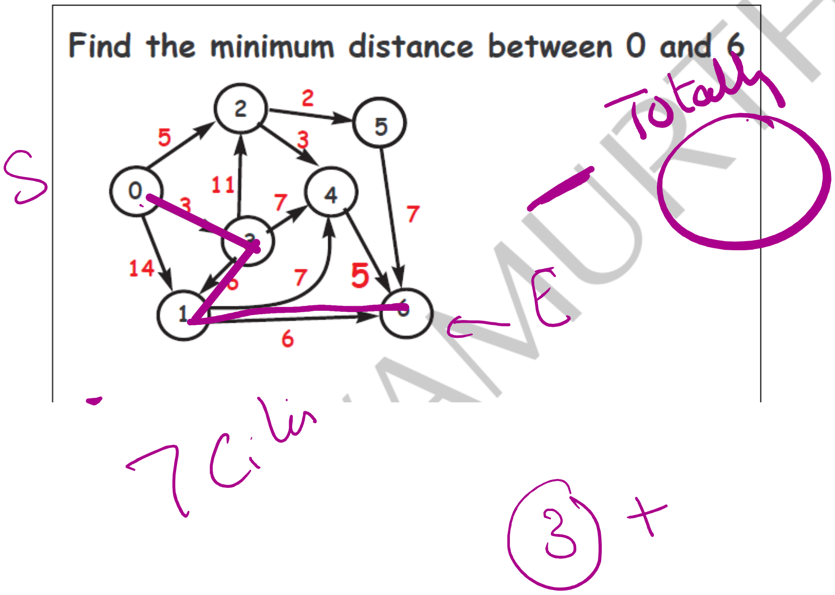
Wednesday, October 18, 2023 5:27 PM

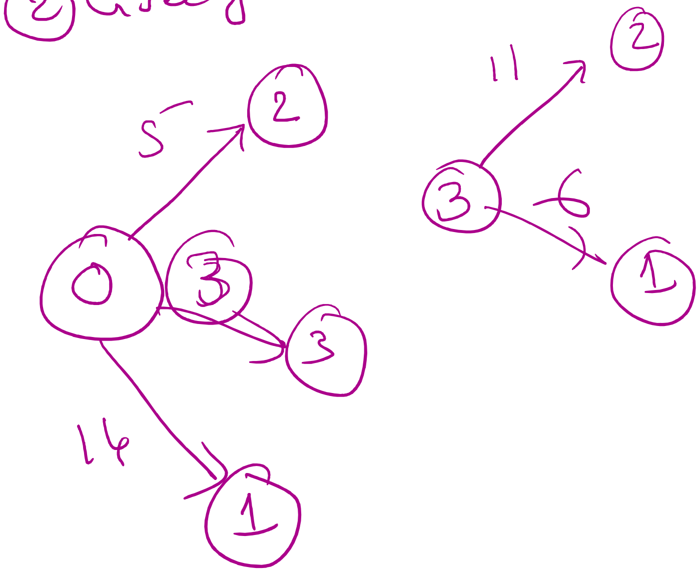
① Brute force

② Greedy D. ..d

Greedy

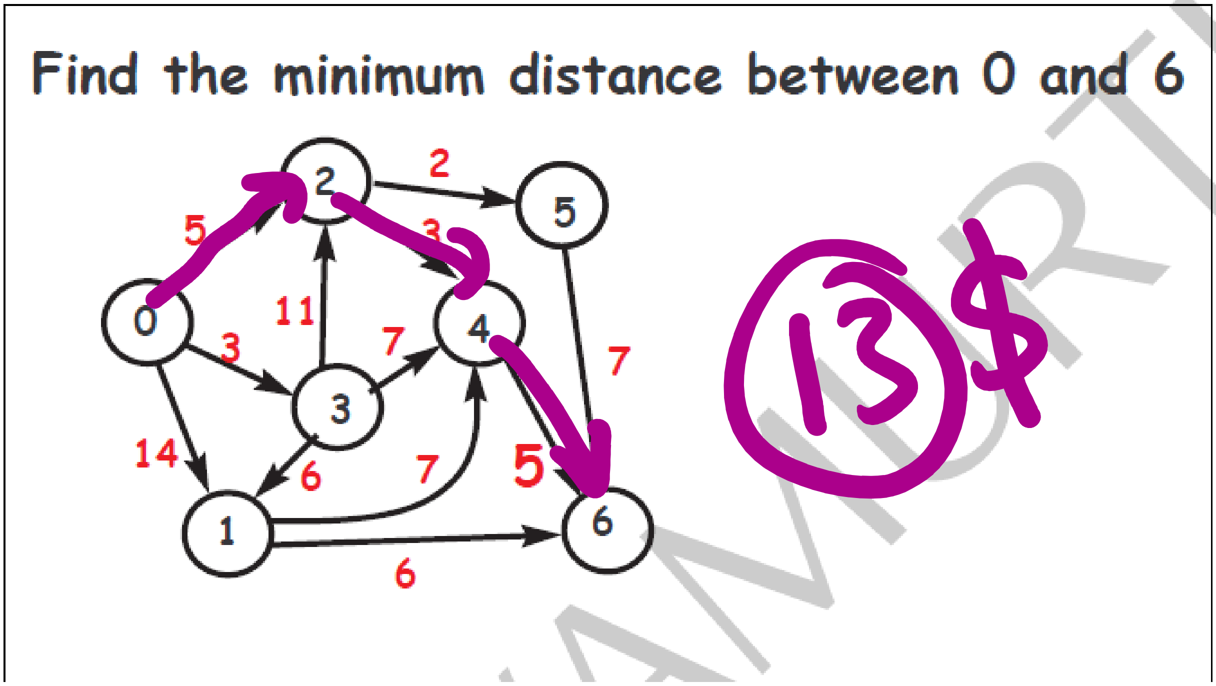
Find the minimum distance between 0 and 6





8

Wednesday, October 18, 2023 5:34 PM



Find the minimum distance between 0 and 6

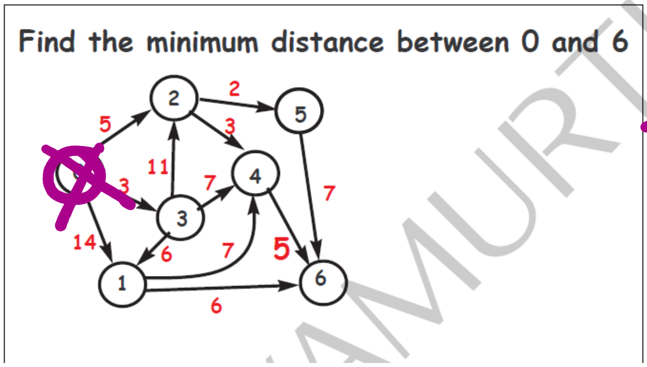
9

Wednesday, October 18, 2023 5:35 PM

1 Course

Topological Sort

Θ(n)



Find the minimum distance between 0 and 6

7 Courses

MS

0,

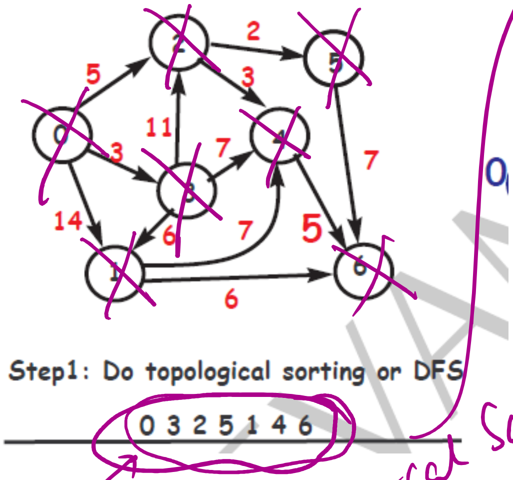
→ 0

10

Wednesday, October 18, 2023 5:38 PM

Dynamic Progra

$\Theta(n)$



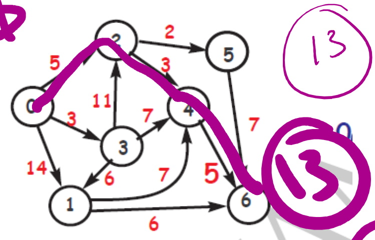
Step1: Do topological sorting or DFS

0 3 2 5 1 4 6

Topological Sort.

Θ(n)

0$



15

Step1: Do topological sorting or DFS

0 3 2 5 1 4 6

Topk

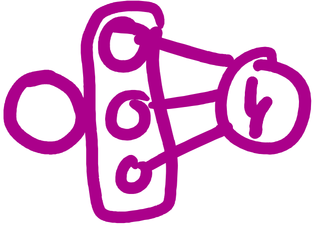
	0	1	2	3	4	5	6
Cost	0	9	5	3	8	7	13
from	∅	3	0	0	2	2	4

SJ X Bos NY

0$

6 → 4 → 2 → 0

0 → 2 → 4 → 6

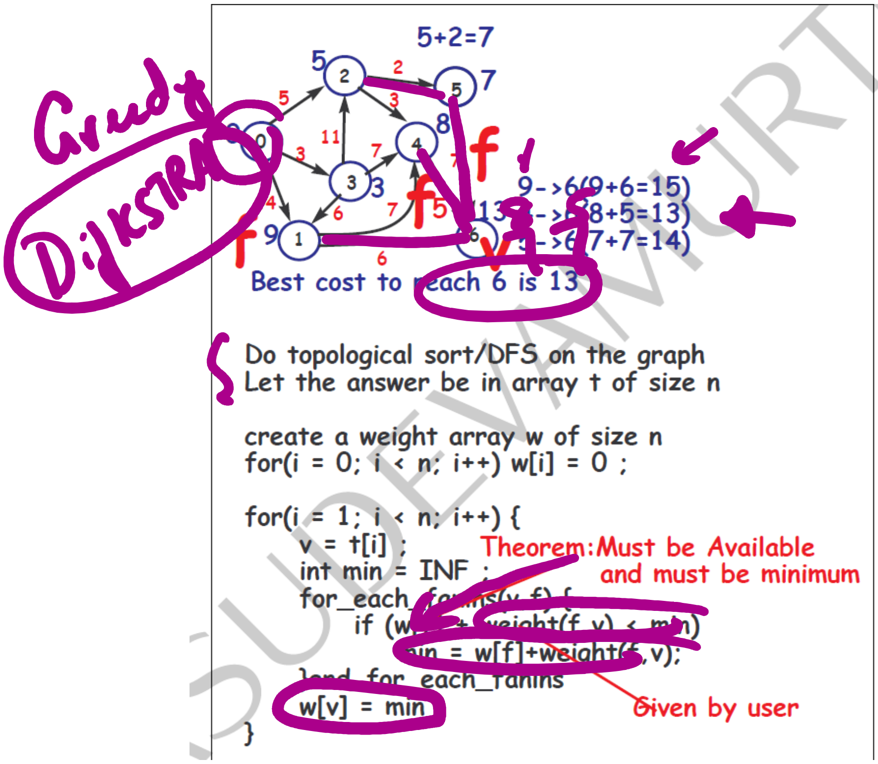


BASE CASE

OPTIMAL Solution For Subproblem

Memoization

OPTIMAL TABLE BUILDING



Best cost to reach 6 is 13

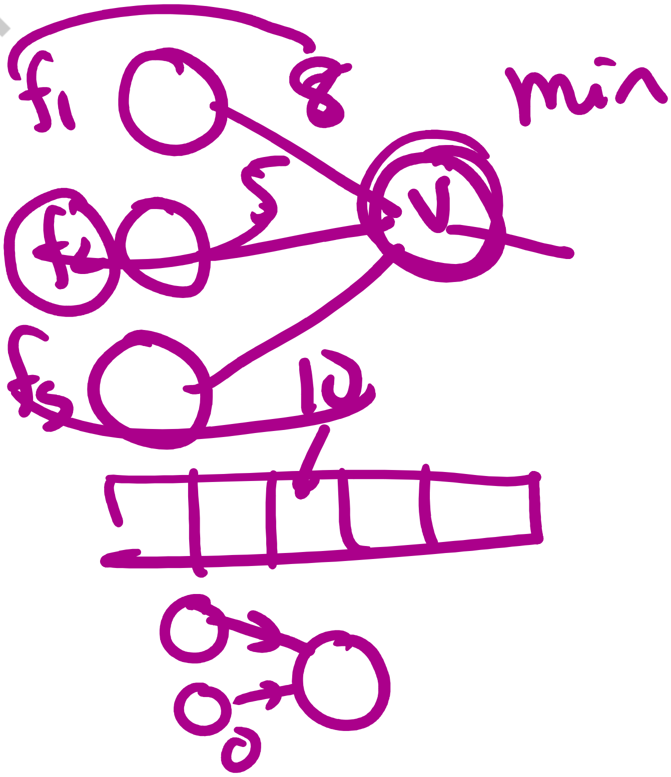
```
Do topological sort/DFS on the graph
Let the answer be in array t of size n

create a weight array w of size n
for(i = 0; i < n; i++) w[i] = 0 ;

for(i = 1; i < n; i++) {
    v = t[i] ;
    int min = INF ;
    for_each_fanins(v,f) {
        if (w[f] + weight(f,v) < min)
            min = w[f]+weight(f,v);
    }end for_each_fanins
    w[v] = min
}
```

Theorem:Must be Available and must be minimum

Given by user



13

Wednesday, October 18, 2023 6:23 PM

Push (x) Θ(1)

x = Pop() — Θ(1)

is full() — Θ(1)

is empty() Θ(1)

x = Top() Θ(1)

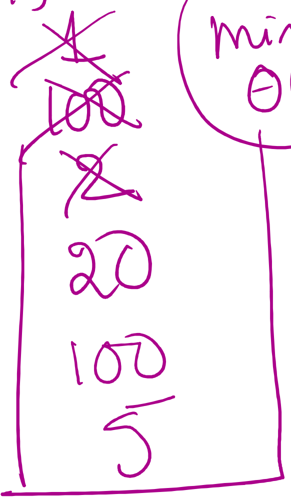
Θ(1)

S List

C Deque

min = Θ(1)

S



min in Θ(1)

min = 2

min = 5

S.min() = None

S.min = 5

5

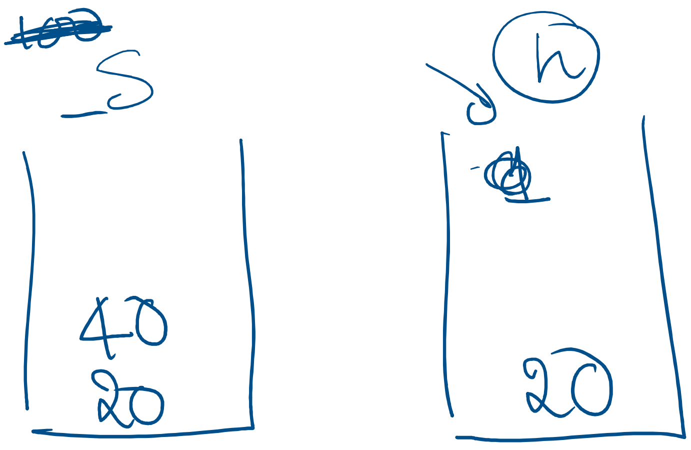
5

min = 2

14

Wednesday, October 18, 2023 6:28 PM

Push Θ(1)
pop
top
isfull
isempty



min

40

top Θ(1)

6:50

HASH

quickSort

2

min Θ(1) time

MAX Θ(1)

15

Wednesday, October 18, 2023 6:51 PM

Insert : Θ(1)

Find Θ(1)

Delete Θ(1)

1 K

K independent of n

n

K 1 Million

a[i]

K ~ 100

HASH

dict

Del

16

Wednesday, October 18, 2023 6:53 PM

[ ] ← Dynai Array

HASH — $\Theta(1)$

$\Theta(1)$

$\Theta(1)$

SLIST

100 Students

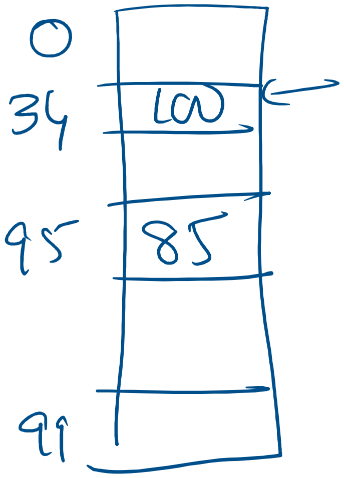
0 .. 99

$\overbrace{408}^{3}\ \overbrace{814}^{3}\ \overbrace{1351}^{4}$

95

Student

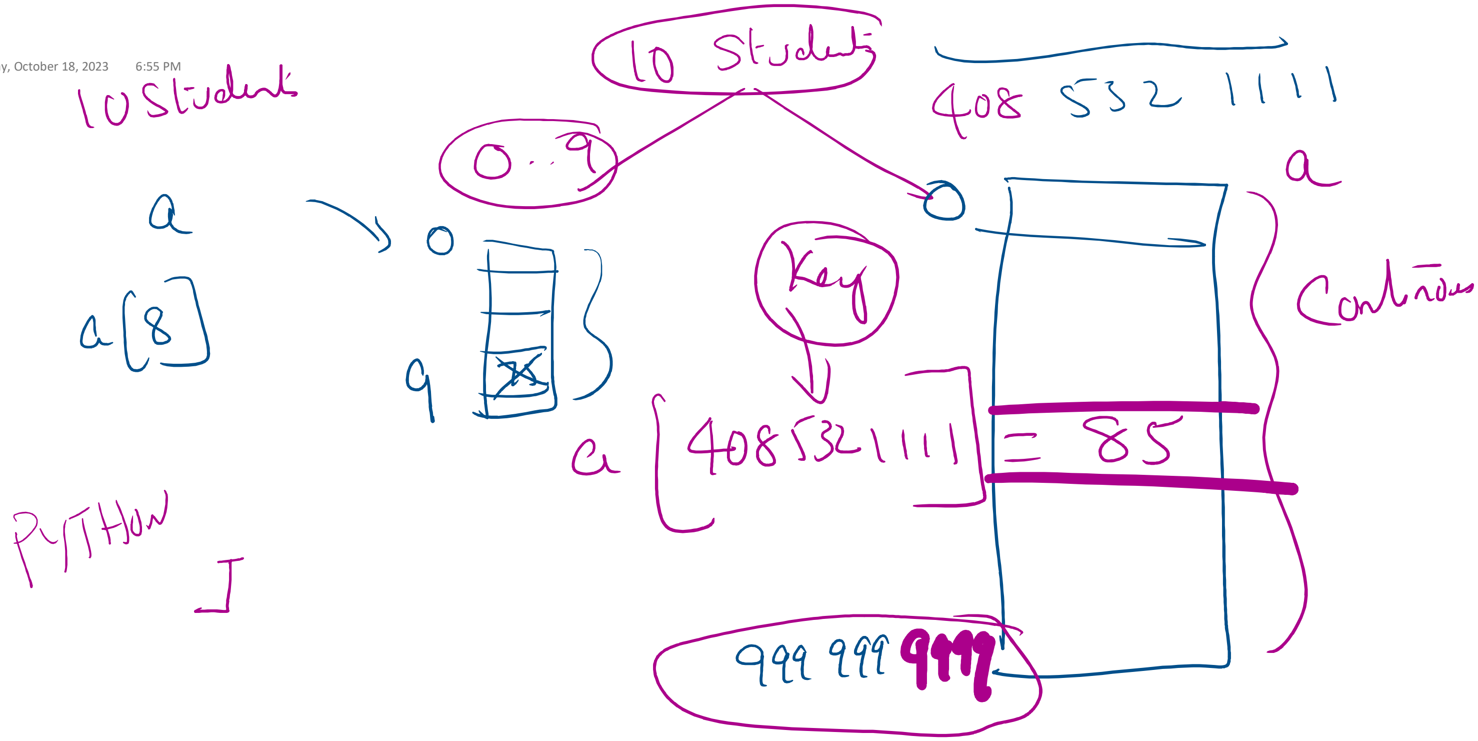
10



$\Theta(1)$ tim

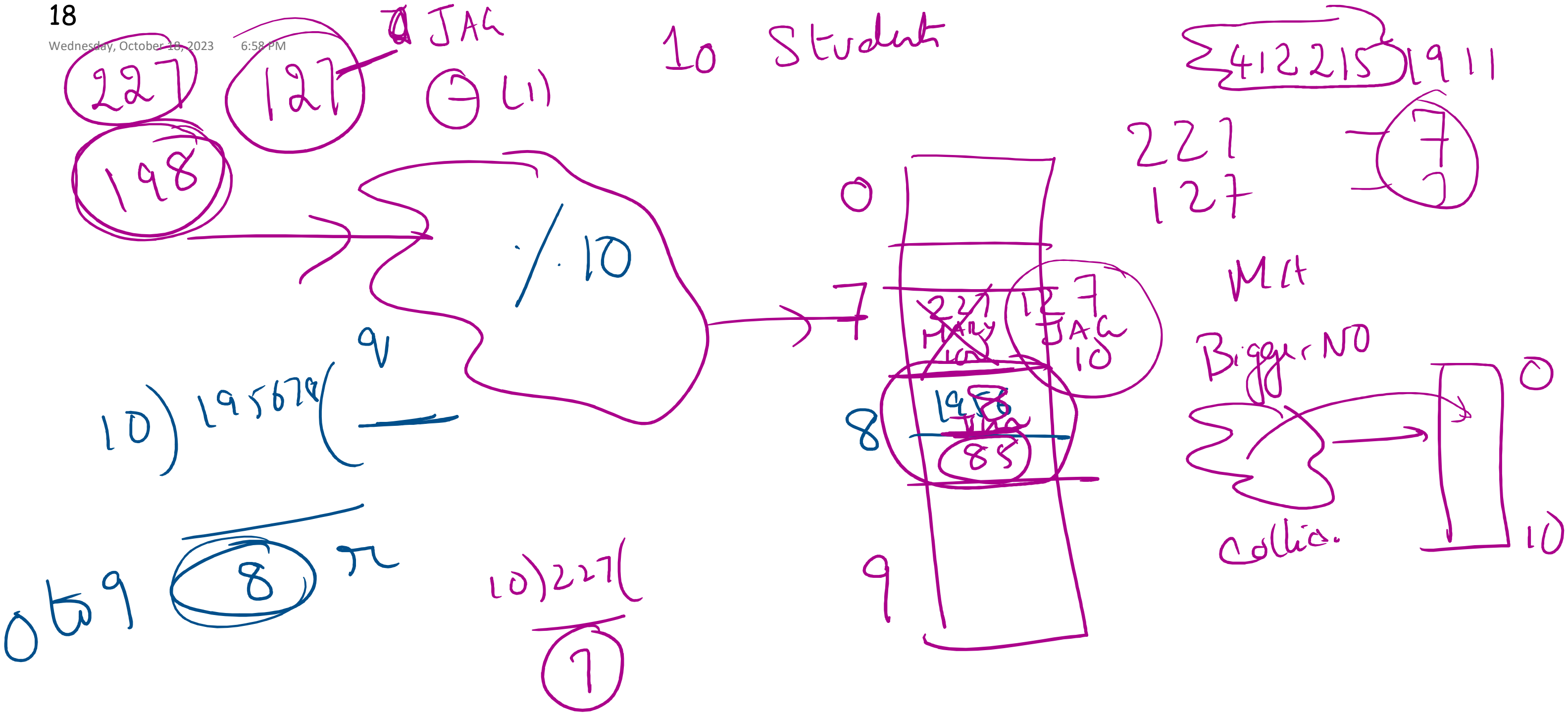
17

Wednesday, October 18, 2023 6:55 PM



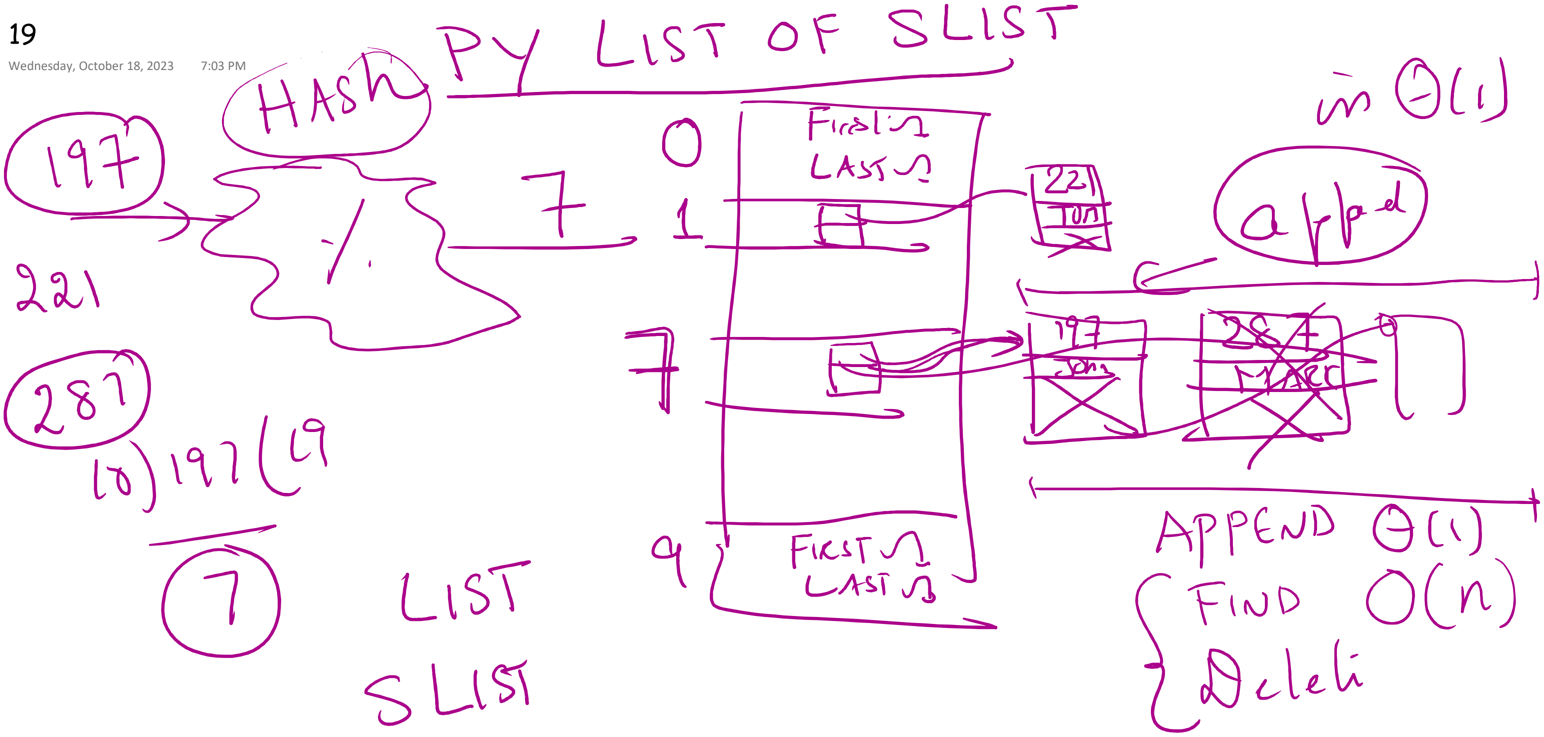
18

Wednesday, October 18, 2023 6:58 PM



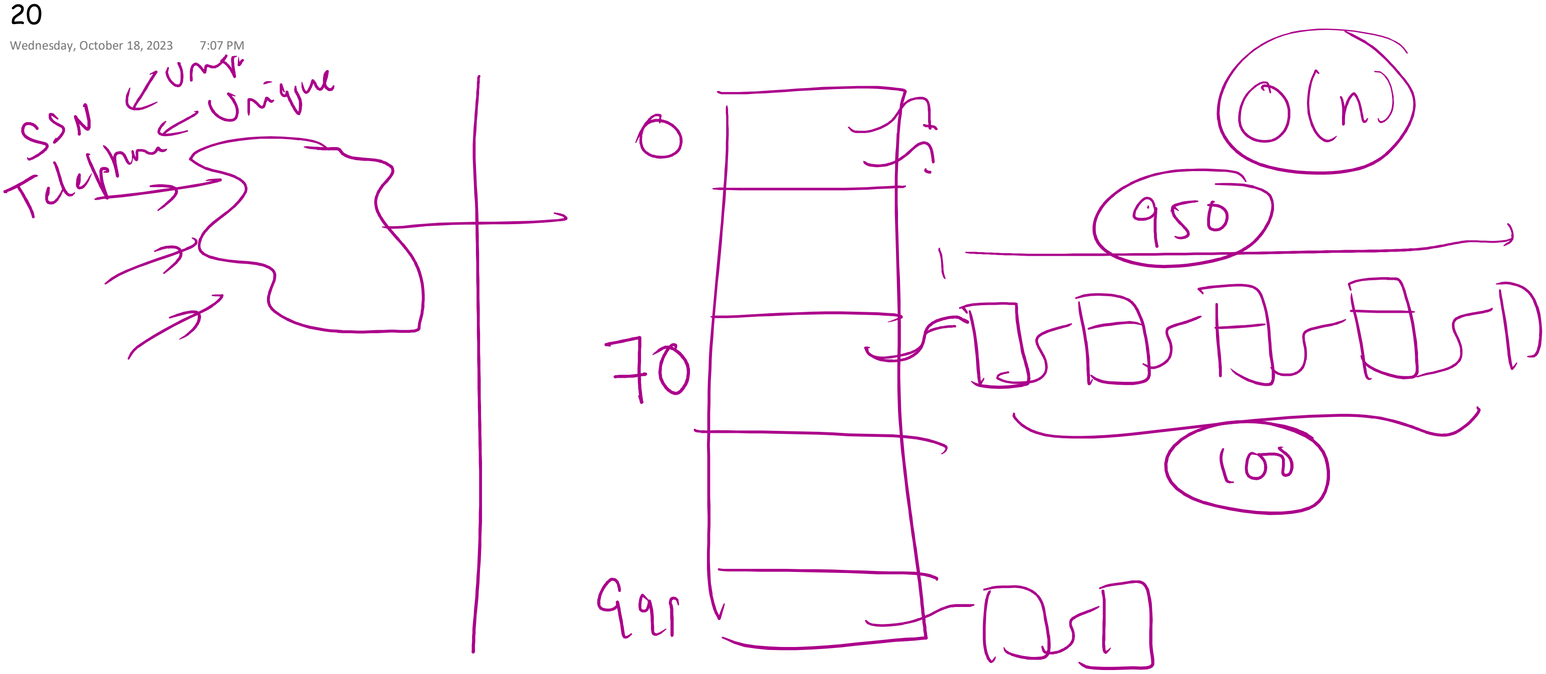
19

Wednesday, October 18, 2023 7:03 PM



20

Wednesday, October 18, 2023 7:07 PM



21

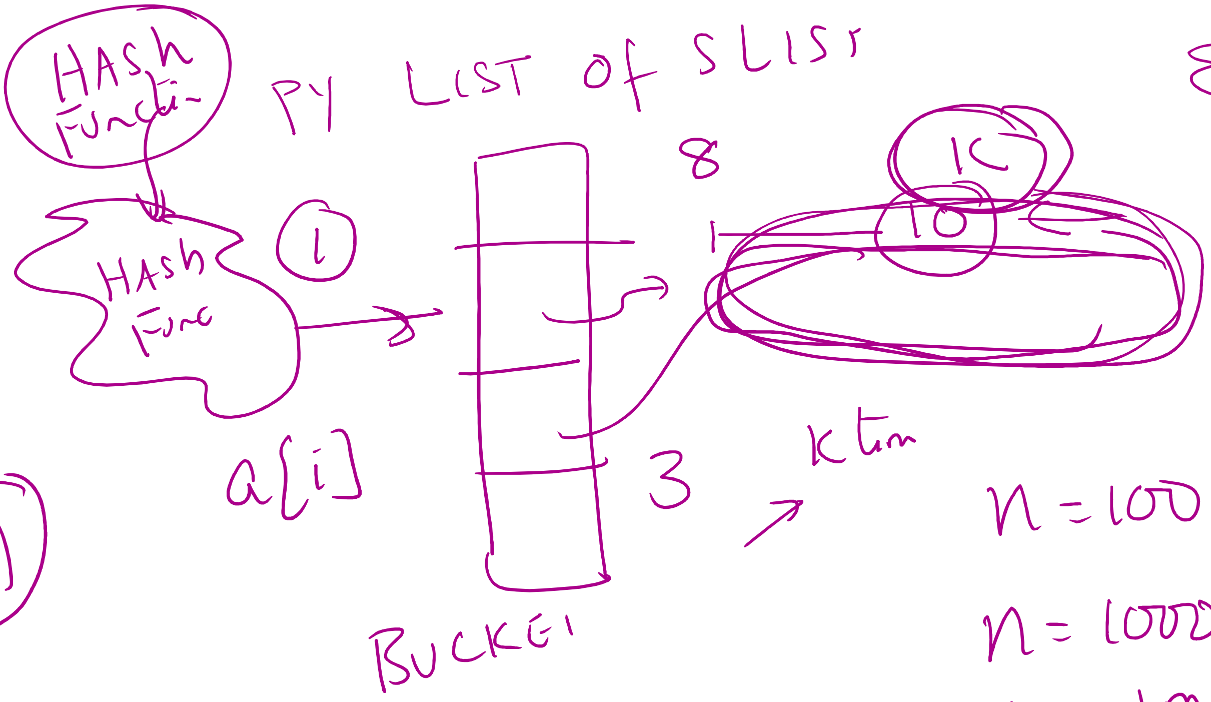
Wednesday, October 18, 2023 7:09 PM

Append

FIND

Delete

$\Theta(1)$



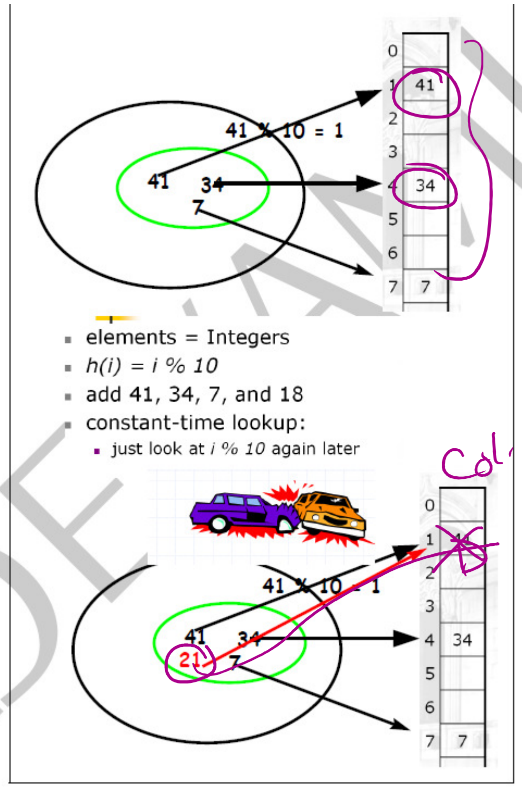
$n = 100$

$n = 10000$

$n = 1$ mill

22

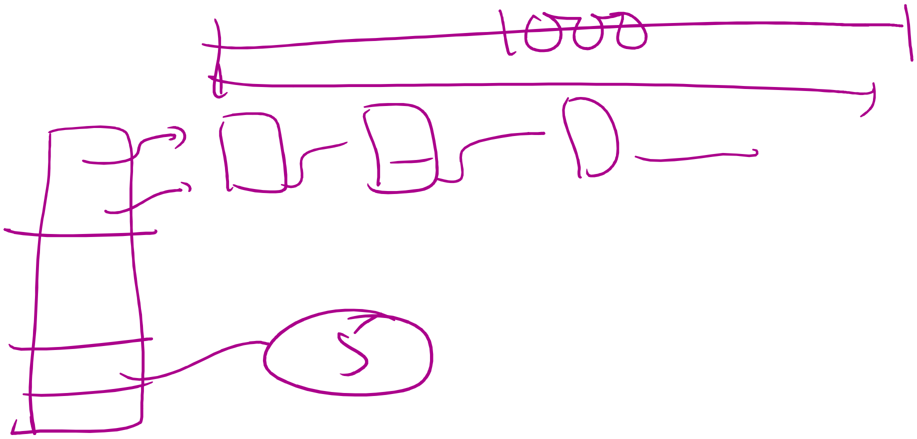
Wednesday, October 18, 2023 7:13 PM

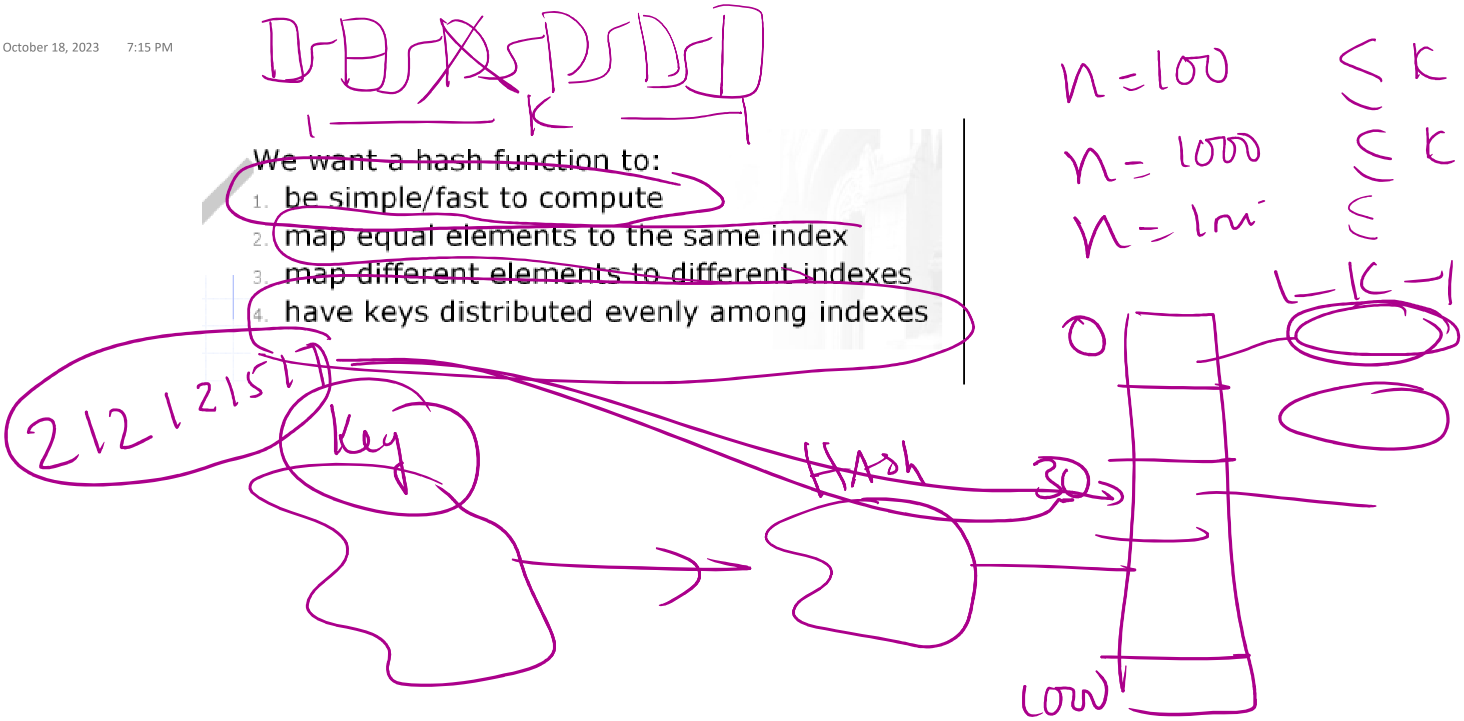


- elements = Integers
- *h(i) = i % 10*
- add 41, 34, 7, and 18
- constant-time lookup:
  - just look at *i % 10* again later

10) 41 (

1



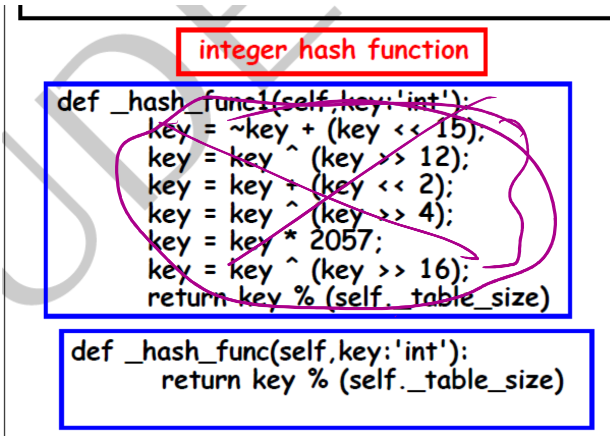


We want a hash function to:

1. be simple/fast to compute
2. map equal elements to the same index
3. ~~map different elements to different indexes~~
4. have keys distributed evenly among indexes

24

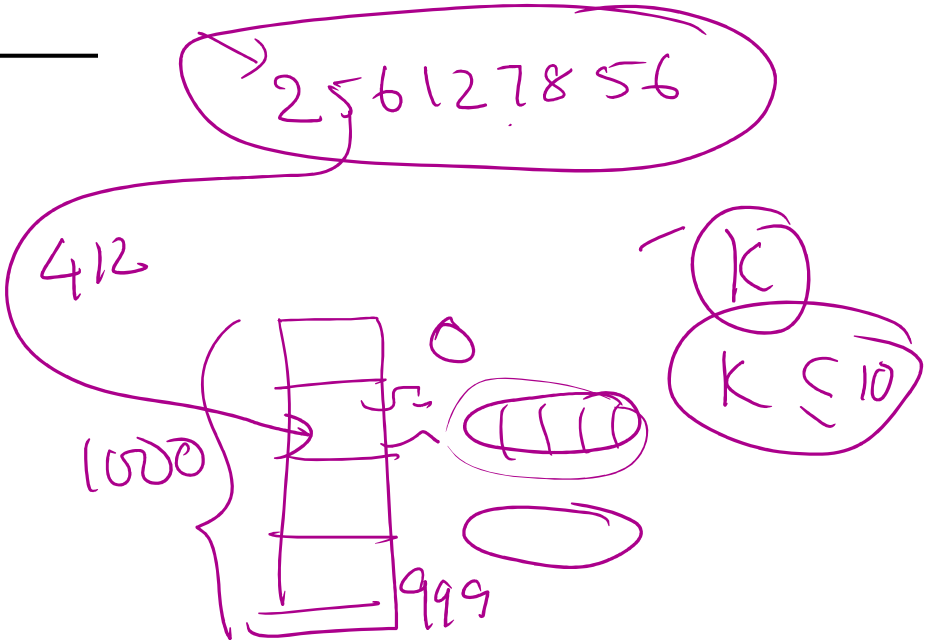
Wednesday, October 18, 2023 7:17 PM



integer hash function

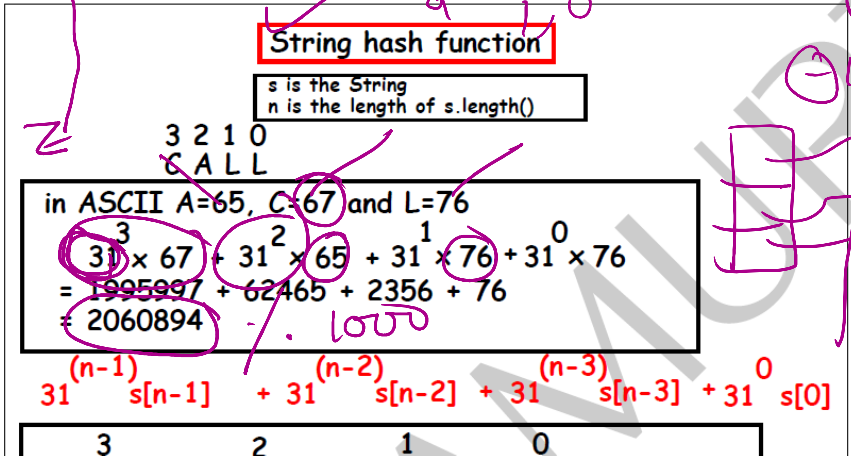
```
def _hash_func1(self,key:'int'):
        key = ~key + (key << 15);
        key = key ^ (key >> 12);
        key = key + (key << 2);
        key = key ^ (key >> 4);
        key = key * 2057;
        key = key ^ (key >> 16);
        return key % (self._table_size)
```

```
def _hash_func(self,key:'int'):
        return key % (self._table_size)
```



A : 65

500 000 ← No. 8



String hash function

s is the String
n is the length of s.length()

3 2 1 0
CALL

in ASCII A=65, C=67 and L=76

$31^3 \times 67 + 31^2 \times 65 + 31^1 \times 76 + 31^0 \times 76$

$= 1995997 + 62465 + 2356 + 76$

$= 2060894$ % 1000

$31^{(n-1)} s[n-1] + 31^{(n-2)} s[n-2] + 31^{(n-3)} s[n-3] + 31^{0} s[0]$

3 2 1 0

3 2 1 0
CALL

ASCII TAD

67

10101111

8

3

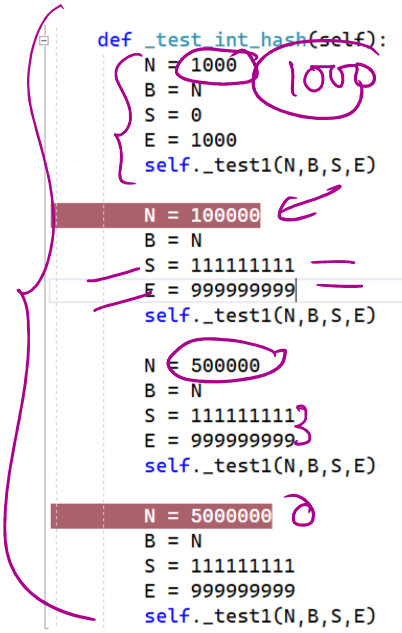
31 *

0

995

26

Wednesday, October 18, 2023 7:24 PM



```
def _test_int_hash(self):
    N = 1000
    B = N
    S = 0
    E = 1000
    self._test1(N,B,S,E)

    N = 100000
    B = N
    S = 111111111
    E = 999999999
    self._test1(N,B,S,E)

    N = 500000
    B = N
    S = 111111111
    E = 999999999
    self._test1(N,B,S,E)

    N = 5000000
    B = N
    S = 111111111
    E = 999999999
    self._test1(N,B,S,E)
```

1000

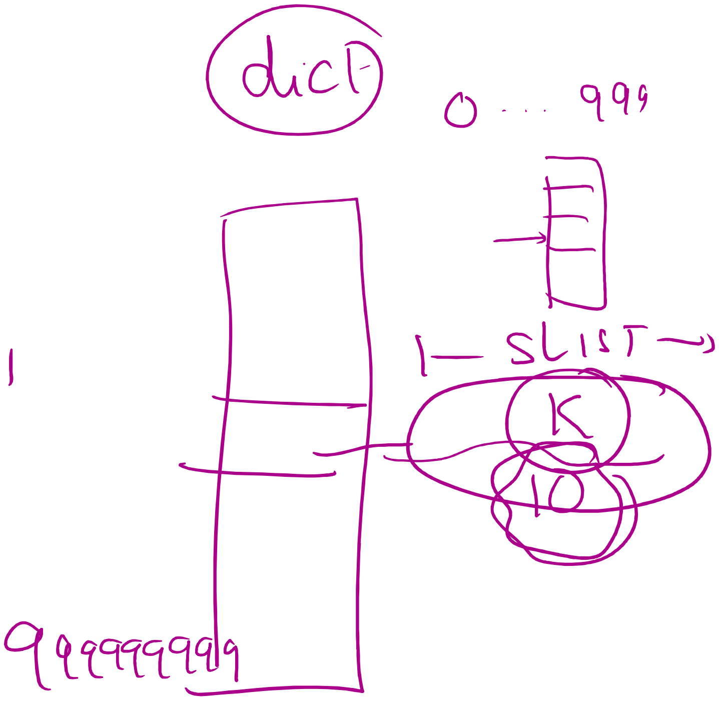
3

0

0..1000

100'000

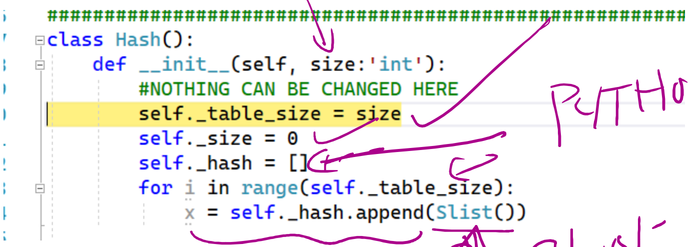
458 256 1271



27

Wednesday, October 18, 2023 7:29 PM

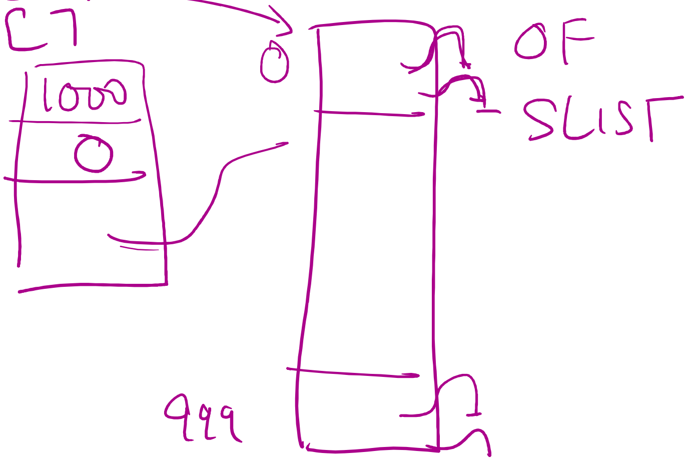
1000 HASH PYTHONLIST



```
##############################################################
class Hash():
    def __init__(self, size:'int'):
        #NOTHING CAN BE CHANGED HERE
        self._table_size = size
        self._size = 0
        self._hash = []
        for i in range(self._table_size):
            x = self._hash.append(Slist())
```

PYTHONLIST

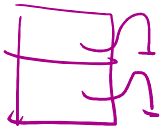
Slist



a = 90

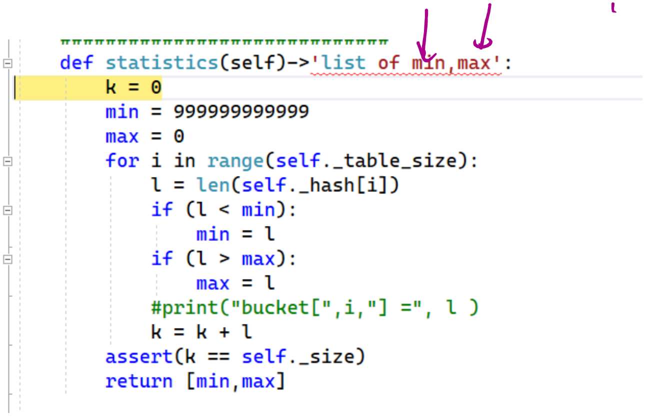
SList S =

S = Slist()

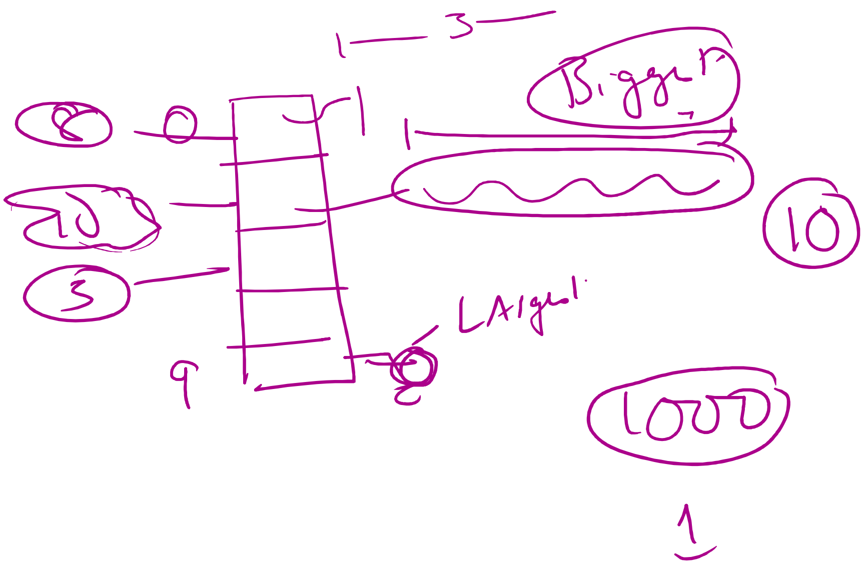


28

Wednesday, October 18, 2023 7:36 PM



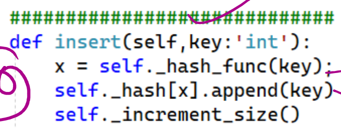
```
    """"""""""""""""""""""""""""""""""""
    def statistics(self)->'list of min,max':
        k = 0
        min = 999999999999
        max = 0
        for i in range(self._table_size):
            l = len(self._hash[i])
            if (l < min):
                min = l
            if (l > max):
                max = l
            #print("bucket[",i,"] =", l )
            k = k + l
        assert(k == self._size)
        return [min,max]
```



29

Wednesday, October 18, 2023 7:43 PM

Θ(1)



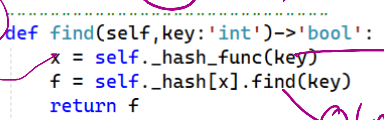
```
################################
def insert(self,key:'int'):
    x = self._hash_func(key);
    self._hash[x].append(key)
    self._increment_size()
```

910

Θ(1)

Θ(1)

Θ(1)

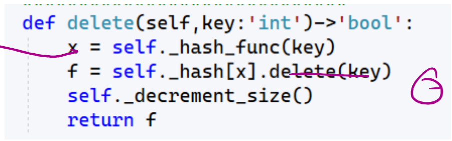


```
def find(self,key:'int')->'bool':
    x = self._hash_func(key)
    f = self._hash[x].find(key)
    return f
```

Θ(n)

Θ(K)

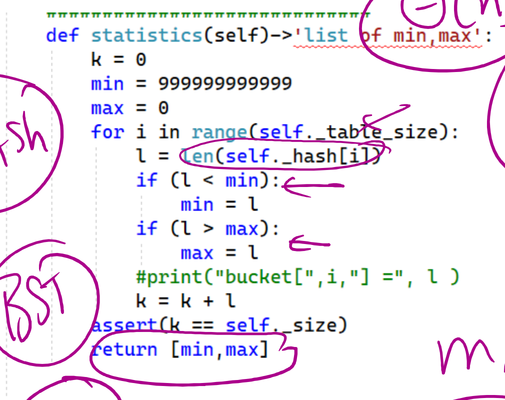
Θ(1)



```
def delete(self,key:'int')->'bool':
    x = self._hash_func(key)
    f = self._hash[x].~~delete(key)~~
    self._decrement_size()
    return f
```

Θ(K)

Θ(n)



```
################################
def statistics(self)->'list of min,max':
    k = 0
    min = 999999999999
    max = 0
    for i in range(self._table_size):
        l = len(self._hash[i])
        if (l < min):
            min = l
        if (l > max):
            max = l
        #print("bucket[",i,"] =", l )
        k = k + l
    assert(k == self._size)
    return [min,max]
```

HASh

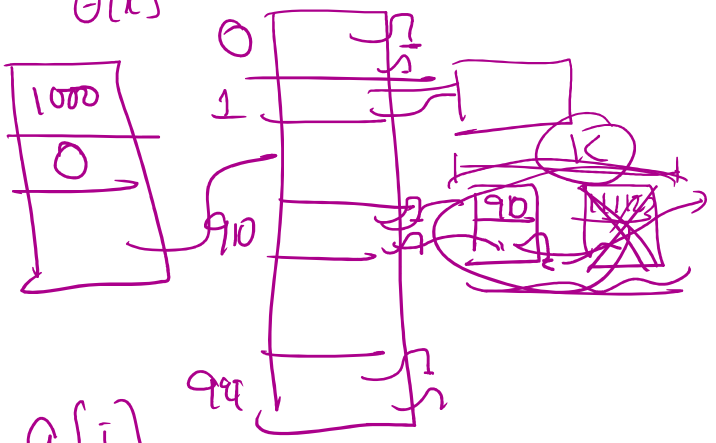
BST

a[i]

n

max ≤ K

Min



a[i]

I7 Page 31

HASh

NO CODE

Dynamic

By HAND

Step by step