DAAPY 10

Wednesday, November 8, 2023 3:50 PM

----------- class 9 11/1/2023 ----------

1 ✓

----------- class 10 11/8/2023 ----------

2

----------- class 11 11/15/2023 ---------- } 2

3

----------- class 12 11/22/2023 ----------

NO CLASS. THANKS GIVING FALL BREAK

----------- class 13 11/29/2023 ----------

4 3

----------- 12/6/2023 ----------

NO CLASS

-------- class 14 12/10/2023 ---

FINAL

1

Wednesday, November 8, 2023 5:03 PM

1million

1

2

Graph

Vertex

tree

Graph

Cost $

miles

G = {Node, edge}

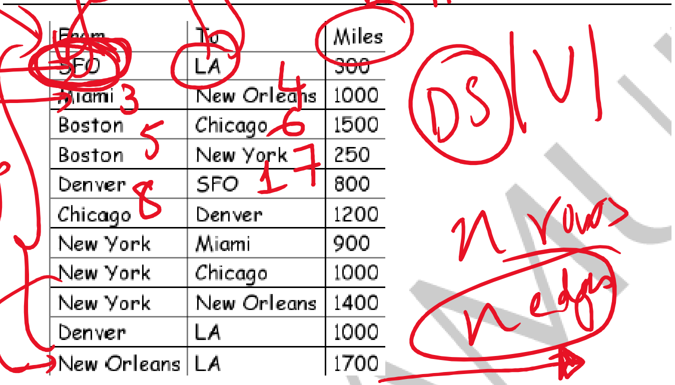
G = {V, {vi, vj}}

DS |V|

Direct

1000 edges

Graph



From	To	Miles
SFO	LA	300
Miami	New Orleans	1000
Boston	Chicago	1500
Boston	New York	250
Denver	SFO	800
Chicago	Denver	1200
New York	Miami	900
New York	Chicago	1000
New York	New Orleans	1400
Denver	LA	1000
New Orleans	LA	1700

3 4 5 6 7 8

n rows

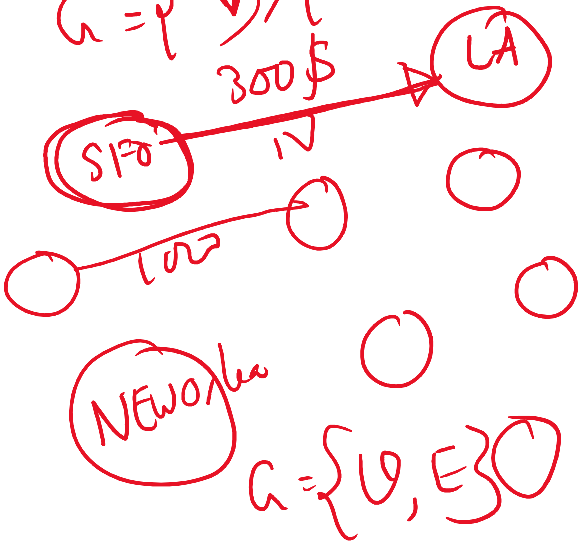
n edges

M

What is the best way to fly from Boston to LA ?

Boston → Miami → LA

1000 miles

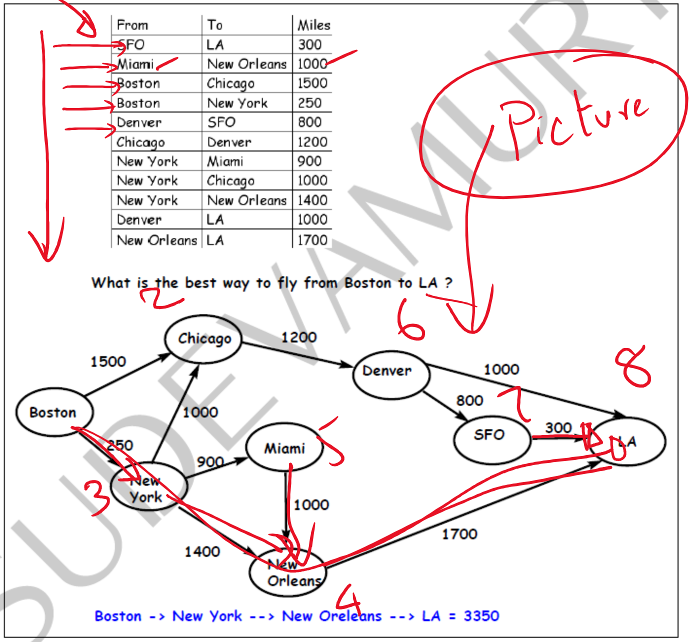


G = {V, E}

2

Wednesday, November 8, 2023 5:11 PM

Memory



From	To	Miles
SFO	LA	300
Miami	New Orleans	1000
Boston	Chicago	1500
Boston	New York	250
Denver	SFO	800
Chicago	Denver	1200
New York	Miami	900
New York	Chicago	1000
New York	New Orleans	1400
Denver	LA	1000
New Orleans	LA	1700

What is the best way to fly from Boston to LA ?

Boston -> New York --> New Oreleans --> LA = 3350

Figure 18.1: What is the best way to fly from Boston to LA?

$G = \{V, E\}$

$\{V, \{V_i, V_j\}\}$ → Undirected

→ directed

$|V| = \boxed{8}$

With Weight

n = 1 millin

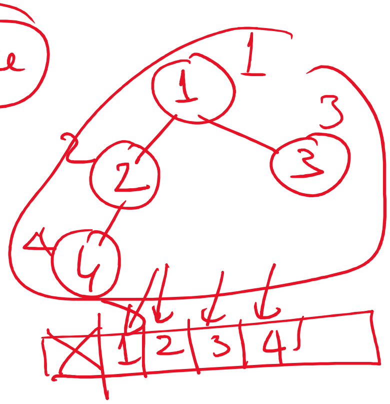
edge = ?

3

Wednesday, November 8, 2023 5:15 PM

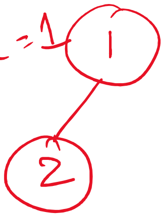
Graph

tree

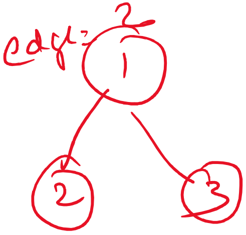


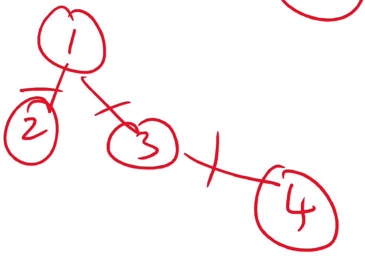
n = 1
edge: 0

n = 2
edge = 1



n = 3
edge: 2



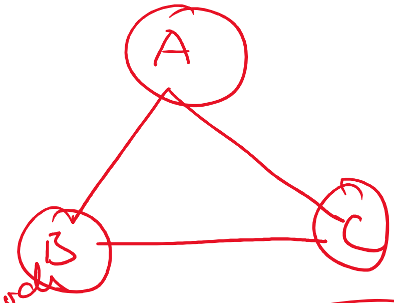


Loop

n nodes
e = (n - 1) edges

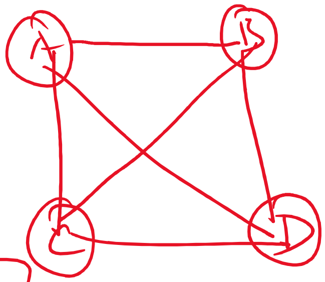
4

Wednesday, November 8, 2023 5:17 PM



Complete graph

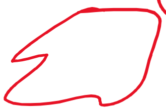
4 nodes



Small

n is very

n = 1 millio



n nodes

$E = \boxed{{}^{n}C_{2}}$ $O(n^2)$

LinkedIn ~~FACEBOOK~~

1 million

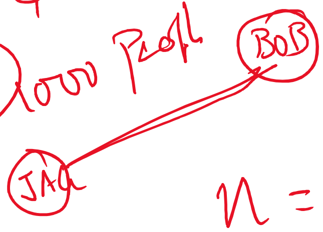
JAG

1 2 3 4

CP

UNION & Find

1000 People



ROB

$n = 1$ million

n

O

1 mi

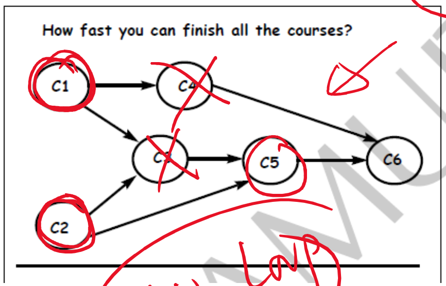
n

TOM

$^{n}C_{2}$

6

Wednesday, November 8, 2023 5:22 PM



How fast you can finish all the courses?

MS

How ?

6 Courses → MS

No Loop

1 Year

1 : C1 & C2

2 : C3, C4

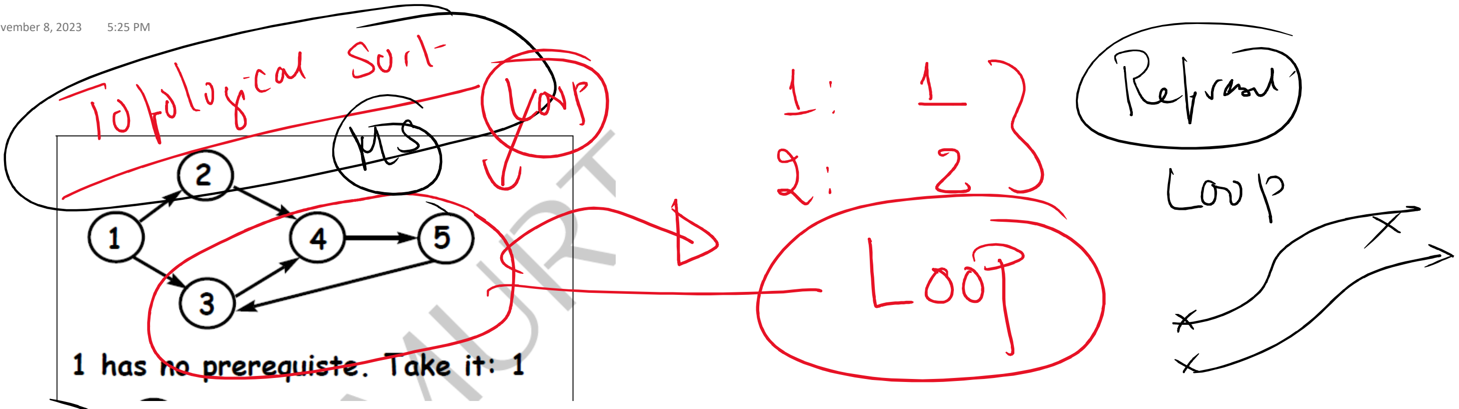
3 : C5

4 : C6

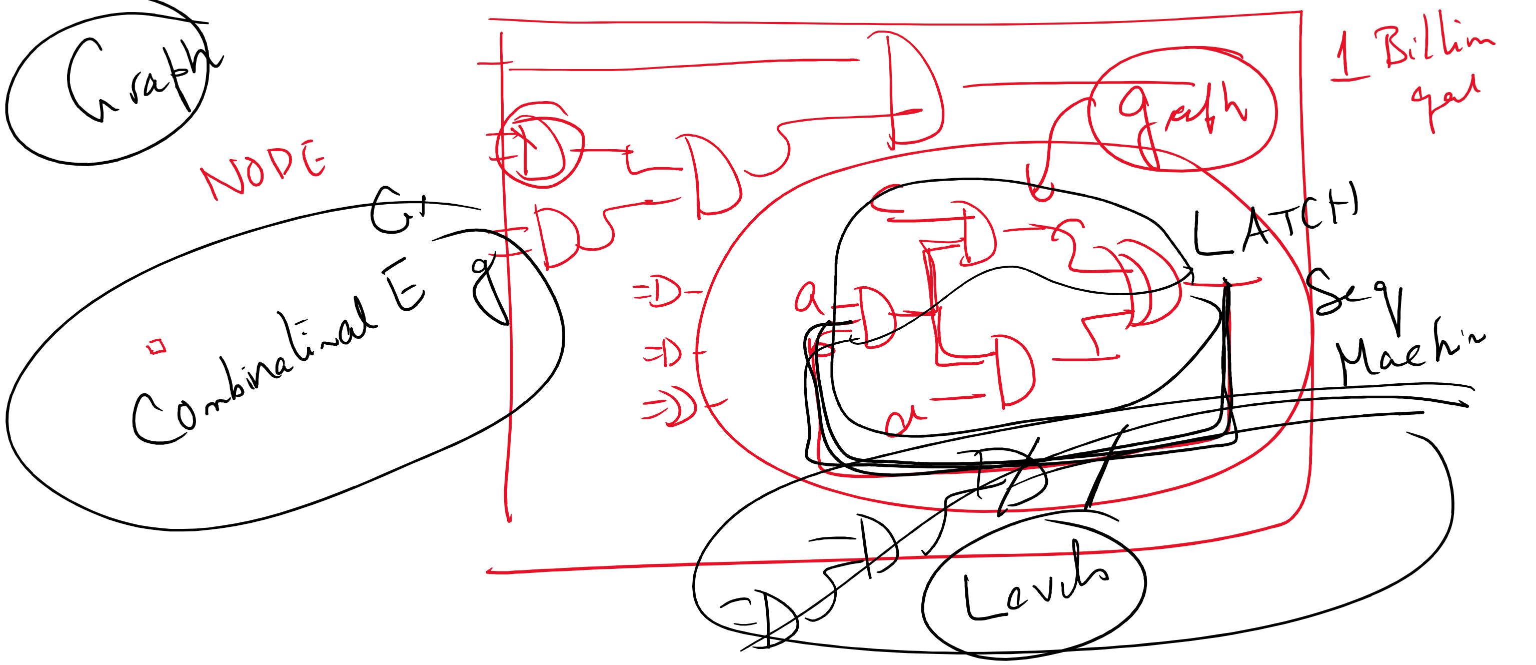
2 Years

1 : C1 & C2

2 .

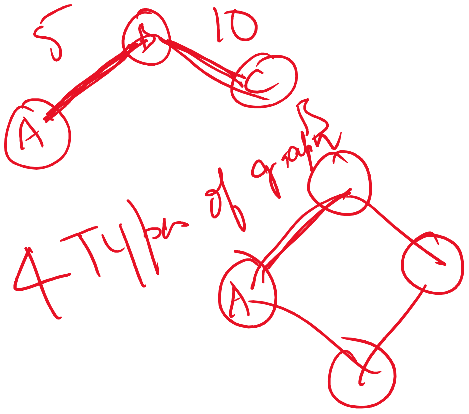


1 has no prerequiste. Take it: 1

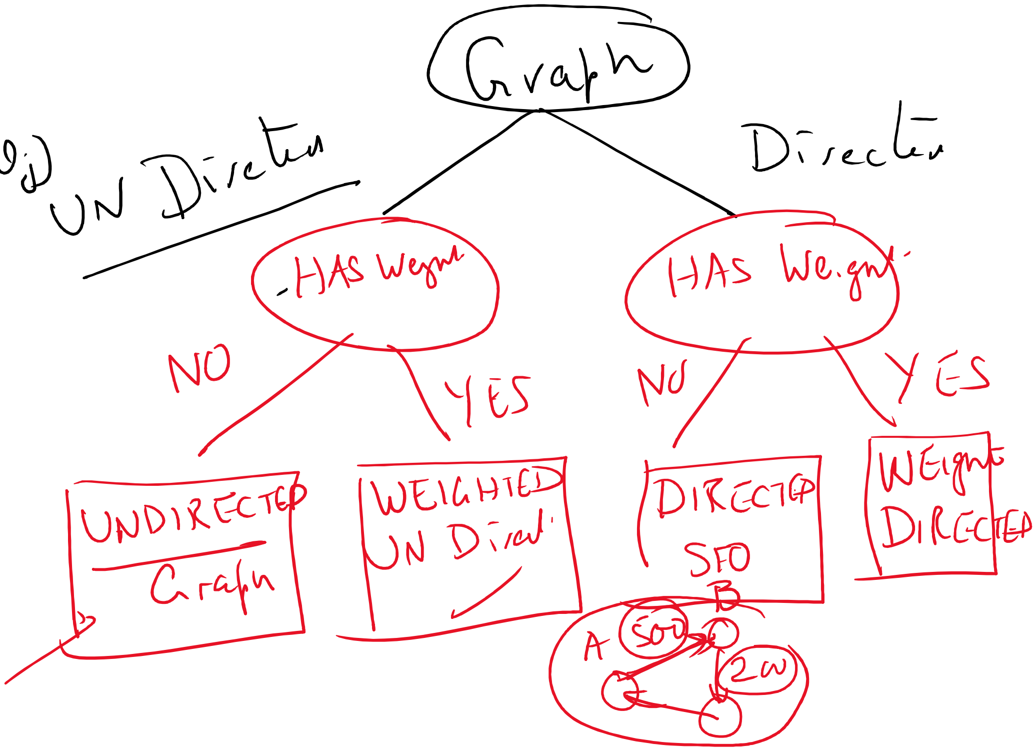


$G = \{V, E\}$

$, \{V_i, V_j\}$

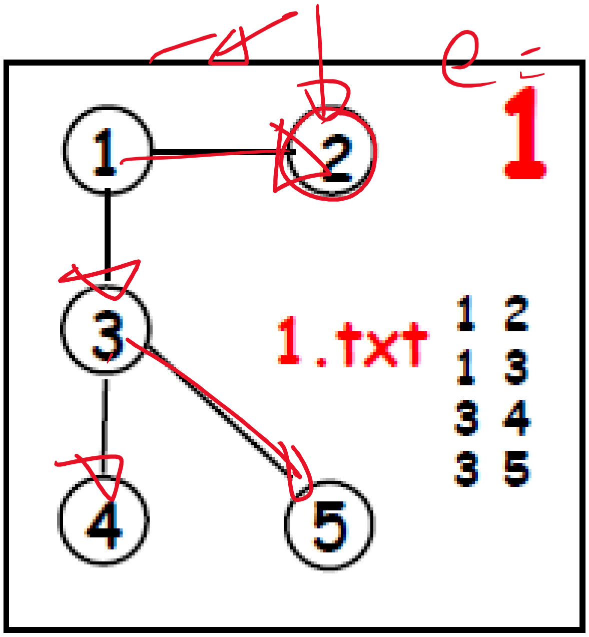


4 Types of graph



9

Wednesday, November 8, 2023 5:34 PM



n = 5

e = 4

① SFO

2 LONDON

Fr, TO

1 2

Graph

1 2

1 3

3 4

3 5

1.txt, Undirected

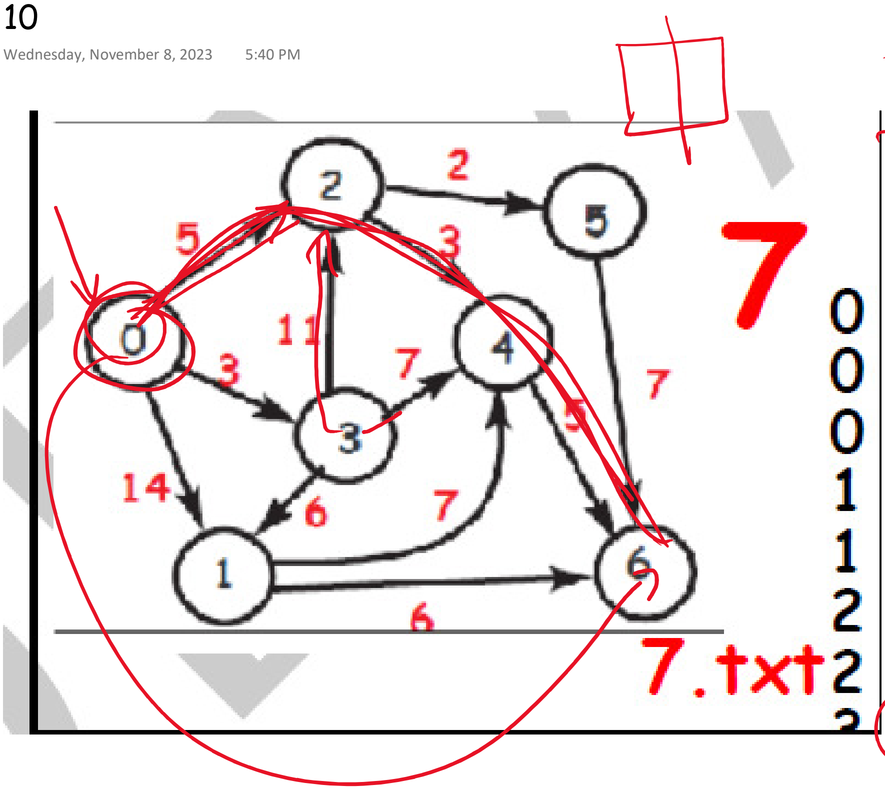
1 2

1 3

1.txt, Dir

10

Wednesday, November 8, 2023 5:40 PM

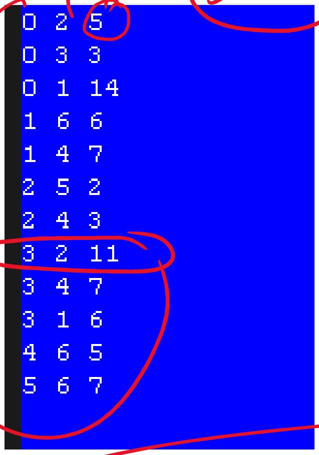


FRom 3col

TO

Cost.

5.2



```
0 2 5
0 3 3
0 1 14
1 6 6
1 4 7
2 5 2
2 4 3
3 2 11
3 4 7
3 1 6
4 6 5
5 6 7
```

Direct

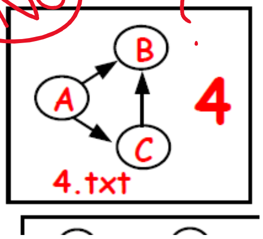
Undir.

4 Types of graph

11

Wednesday, November 8, 2023 5:42 PM

No ?

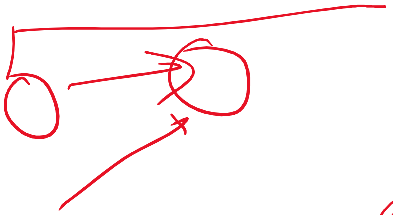


4

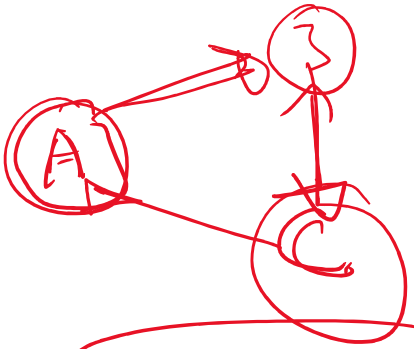
A B

A C

C B



10,000$



Direct Graph With Loop

No Loop

Direct graph With No Loop

DAG

Direct Acyclic Graph

DAG

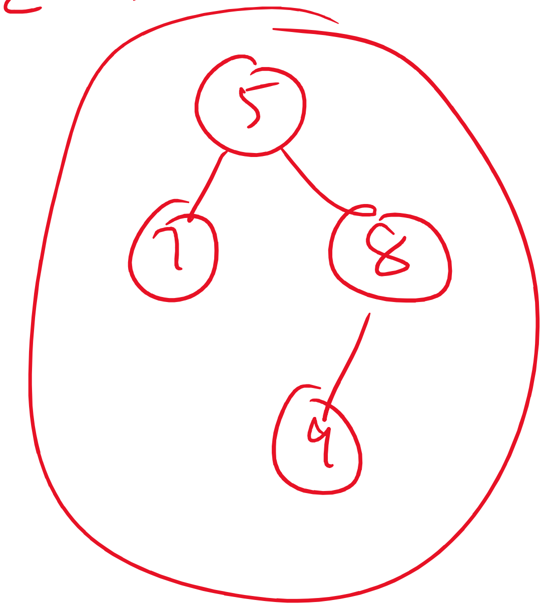
12

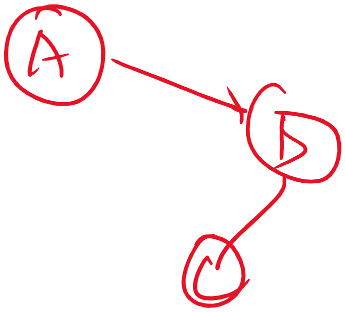
Wednesday, November 8, 2023 5:48 PM

[1, 2, 3, 10, 75]

1 → 2 → 10 → 75 → NONE

GRAPHVIZ



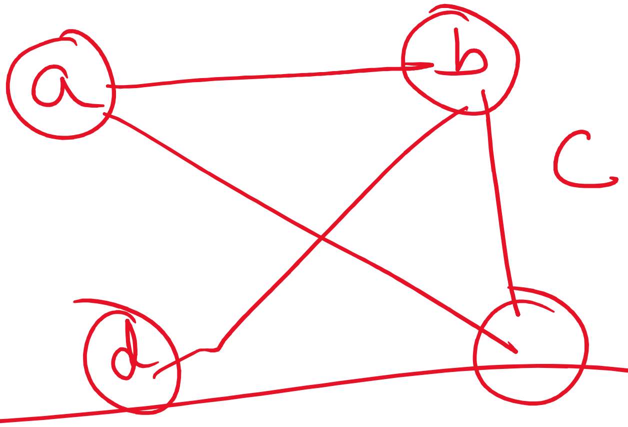


13

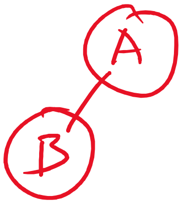
Wednesday, November 8, 2023 5:51 PM

```
graph graphname {
   a -- b;
   b -- c;
   b -- d;
   d -- a;
}
```

-- -



	1	2	3	4
	A	B	D	E



A — B
A — D
B — E

14

Wednesday, November 8, 2023 6:01 PM

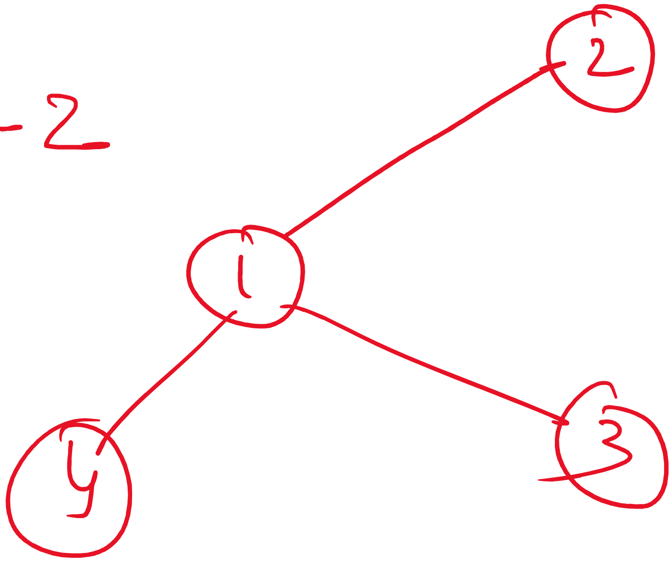
g

```
## Jagadeesh Vasudevamurthy ####
diagraph g {
        edge [dir=none, color=red]
          1 -> 2
          1 -> 3
          1 -> 4
          2 -> 5
          2 -> 4
          3 -> 6
          3 -> 4
          4 -> 5
          4 -> 7
          4 -> 6
          5 -> 7
          6 -> 7
}
```

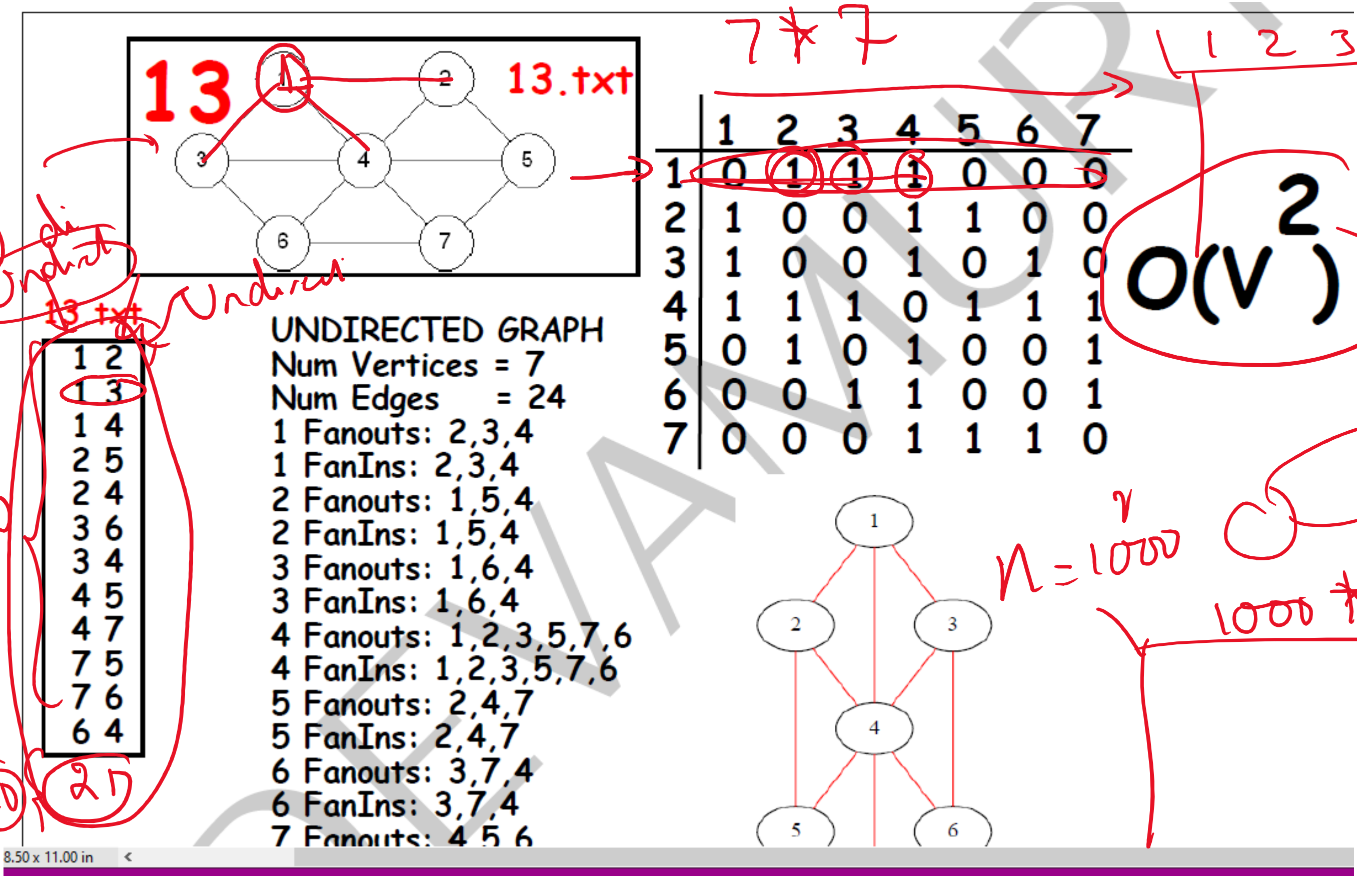
D

1 — 2

1



Wednesday, November 8, 2023 6:08 PM



13.txt

1	2
1	3
1	4
2	5
2	4
3	6
3	4
4	5
4	7
7	5
7	6
6	4

UNDIRECTED GRAPH
Num Vertices = 7
Num Edges = 24
1 Fanouts: 2,3,4
1 FanIns: 2,3,4
2 Fanouts: 1,5,4
2 FanIns: 1,5,4
3 Fanouts: 1,6,4
3 FanIns: 1,6,4
4 Fanouts: 1,2,3,5,7,6
4 FanIns: 1,2,3,5,7,6
5 Fanouts: 2,4,7
5 FanIns: 2,4,7
6 Fanouts: 3,7,4
6 FanIns: 3,7,4
7 Fanouts: 4,5,6

$$\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 3 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 4 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 5 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 6 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 7 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}$$

$O(V^2)$

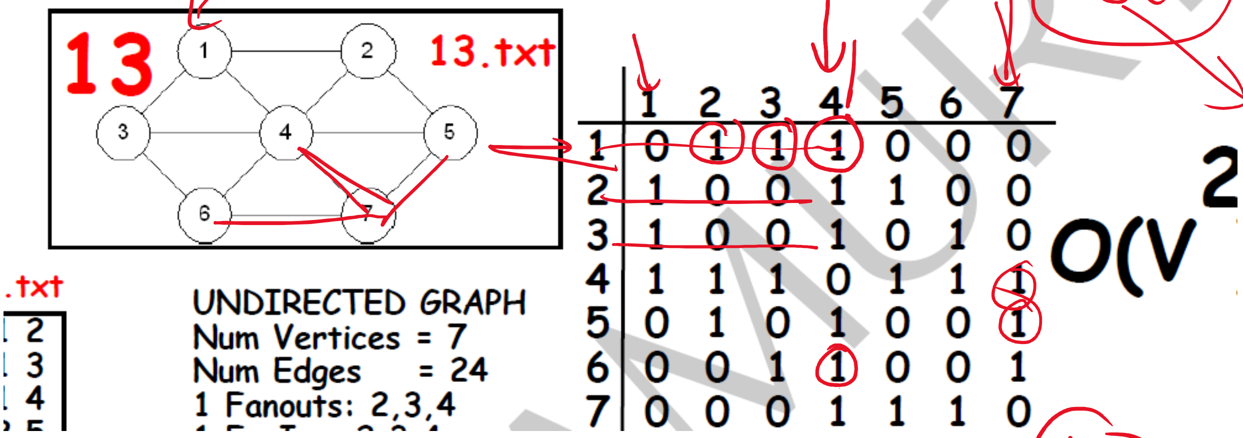
①

SPACE: $O(V^2)$

1 Bill

$O(V)$

$\sqrt{O(V)}$



13

13.txt

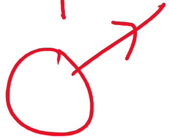
.txt

2
3
4

UNDIRECTED GRAPH
Num Vertices = 7
Num Edges = 24
1 Fanouts: 2,3,4

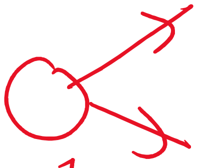
$$\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 3 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 4 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 5 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 6 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 7 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}$$

$O(V^2$



Where CAN I go from ① $O(V)$

$O(V)$



$$\begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 10 & N \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ & 0 & 0 & 1 & 0 & & & \end{array}$$

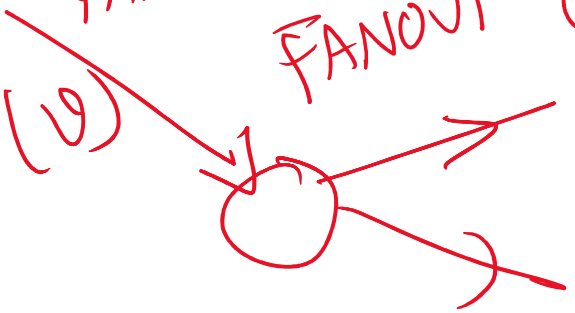
② SPARSE MATRIX

FANIN

FANOUT ① $O(V^2)$

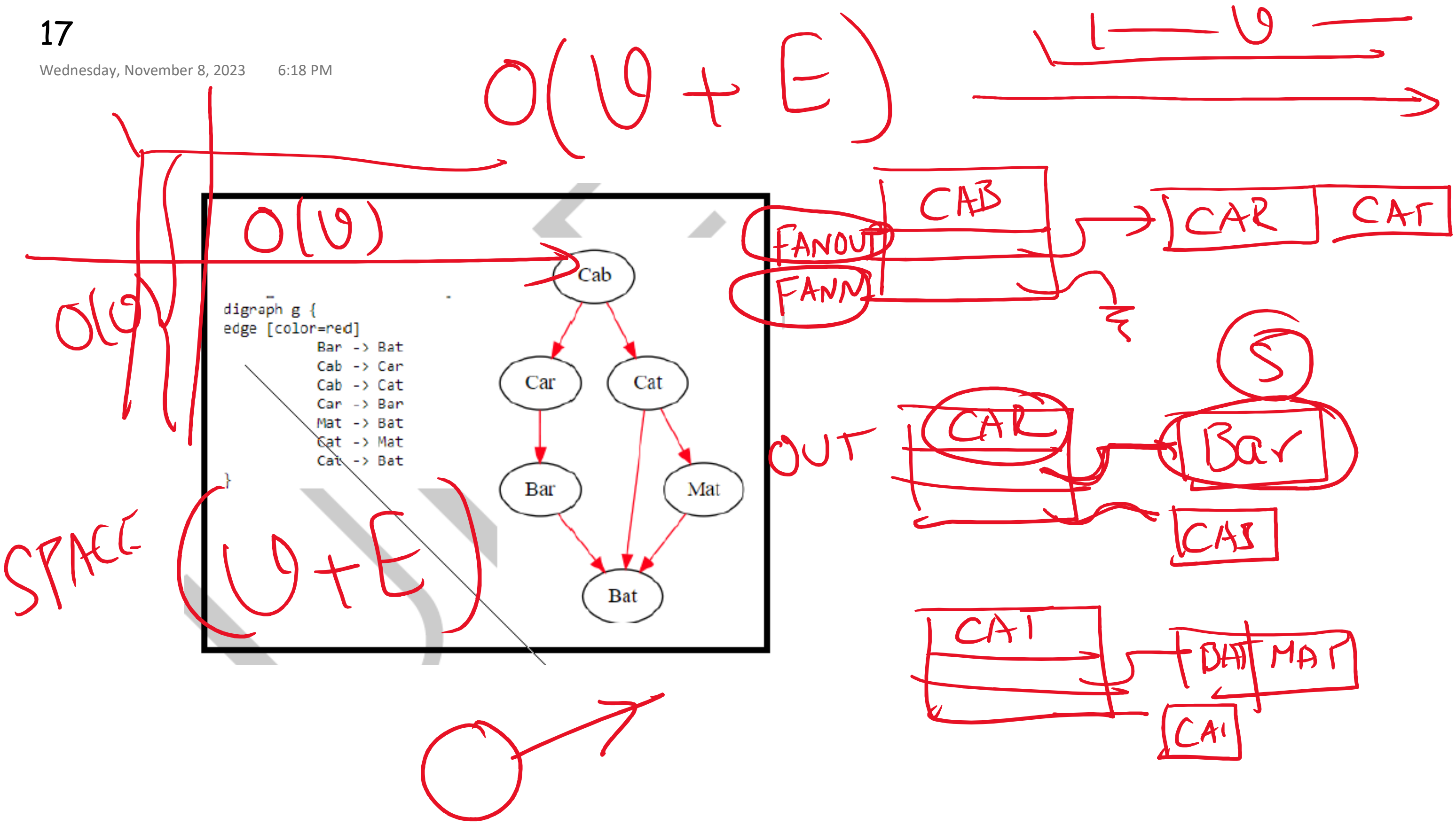
$\boxed{O(V+E)}$

$O(V)$



$O(V)$ Operati

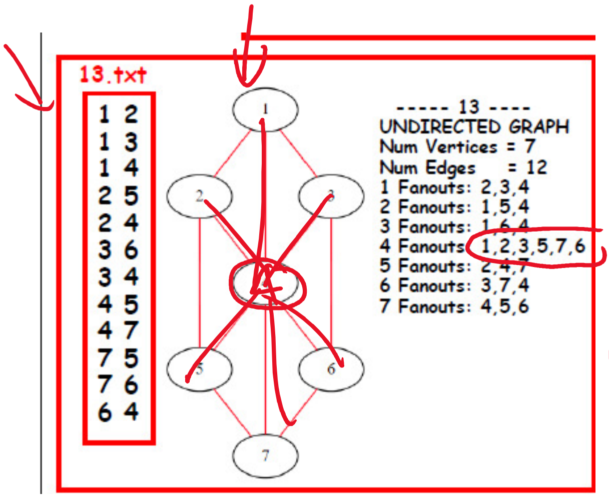
Wednesday, November 8, 2023 6:18 PM



```
digraph g {
edge [color=red]
        Bar  -> Bat
        Cab  -> Car
        Cab  -> Cat
        Car  -> Bar
        Mat  -> Bat
        Cat  -> Mat
        Cat  -> Bat
}
```

18

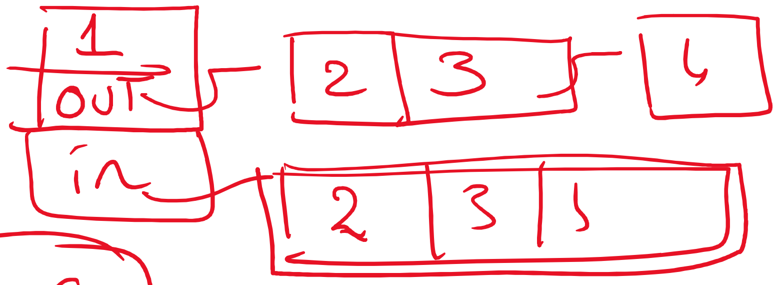
Wednesday, November 8, 2023 6:22 PM



13.txt

```
1 2
1 3
1 4
2 5
2 4
3 6
3 4
4 5
4 7
7 5
7 6
6 4
```

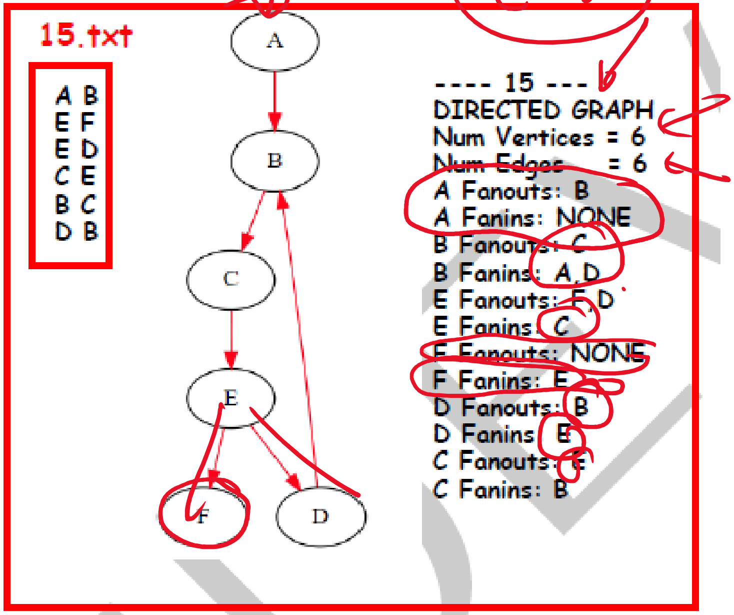
```
 ----- 13 ----
UNDIRECTED GRAPH
Num Vertices = 7
Num Edges    = 12
1 Fanouts: 2,3,4
2 Fanouts: 1,5,4
3 Fanouts: 1,6,4
4 Fanouts: 1,2,3,5,7,6
5 Fanouts: 2,4,7
6 Fanouts: 3,7,4
7 Fanouts: 4,5,6
```



V+E

Wednesday, November 8, 2023 6:24 PM

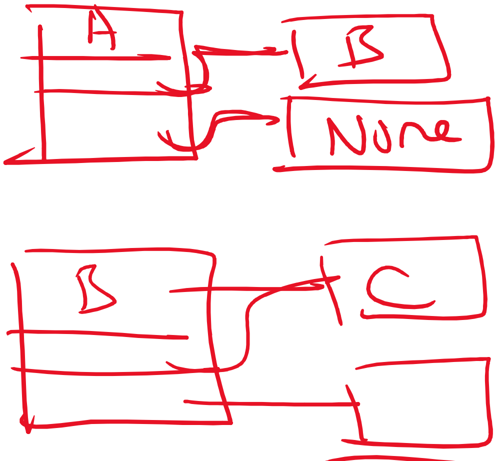
E+V



15.txt

```
A B
E F
E D
C E
B C
D B
```

```
---- 15 ---
DIRECTED GRAPH
Num Vertices = 6
Num Edges    = 6
A Fanouts: B
A Fanins: NONE
B Fanouts: C
B Fanins: A,D
E Fanouts: F,D
E Fanins: C
F Fanouts: NONE
F Fanins: E
D Fanouts: B
D Fanins: E
C Fanouts: E
C Fanins: B
```



O(V)

O(1)

20

Wednesday, November 8, 2023 6:26 PM

2

6:50

NetworkX

10

π Short-path

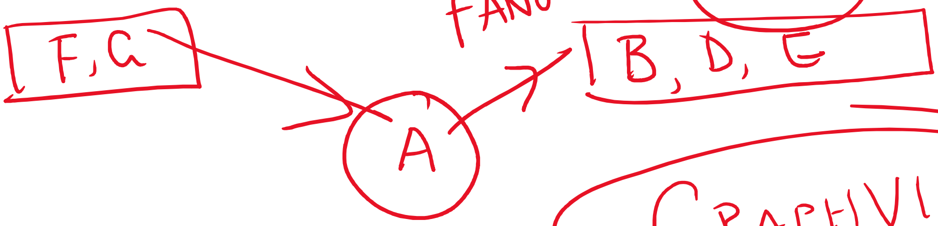
GRAPH

1 $O(V+E)$

$O(V^2)$

FANOUT $\Theta(1)$

2



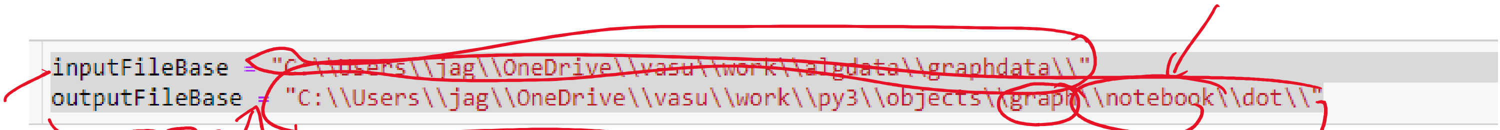
GRAPHVIZ

21

Wednesday, November 8, 2023 6:54 PM

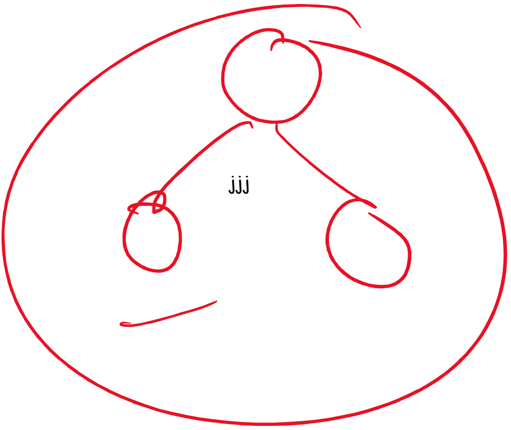
13

13.txt



```
inputFileBase = "C:\\Users\\jag\\OneDrive\\vasu\\work\\algdata\\graphdata\\"
outputFileBase = "C:\\Users\\jag\\OneDrive\\vasu\\work\\py3\\objects\\graph\\notebook\\dot\\"
```

.dot

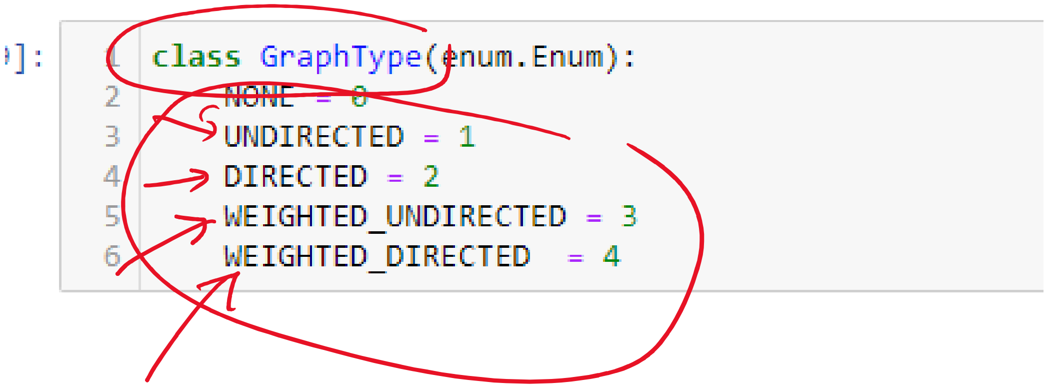


2

22

Wednesday, November 8, 2023 7:00 PM

13, ~~UNDIRECD~~ Direct

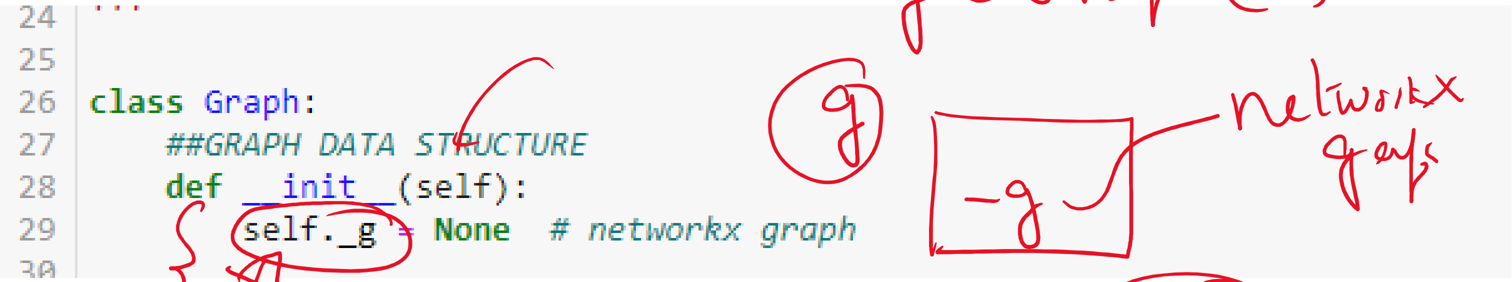


Graph Types

```
]:
class GraphType(enum.Enum):
    ~~NONE = 0~~
    UNDIRECTED = 1
    DIRECTED = 2
    WEIGHTED_UNDIRECTED = 3
    WEIGHTED_DIRECTED   = 4
```

23

Wednesday, November 8, 2023 7:01 PM



```
    '''

class Graph:
    ##GRAPH DATA STRUCTURE
    def __init__(self):
        self._g = None  # networkx graph
```

g = Graph( )

g

_g

networkx graph

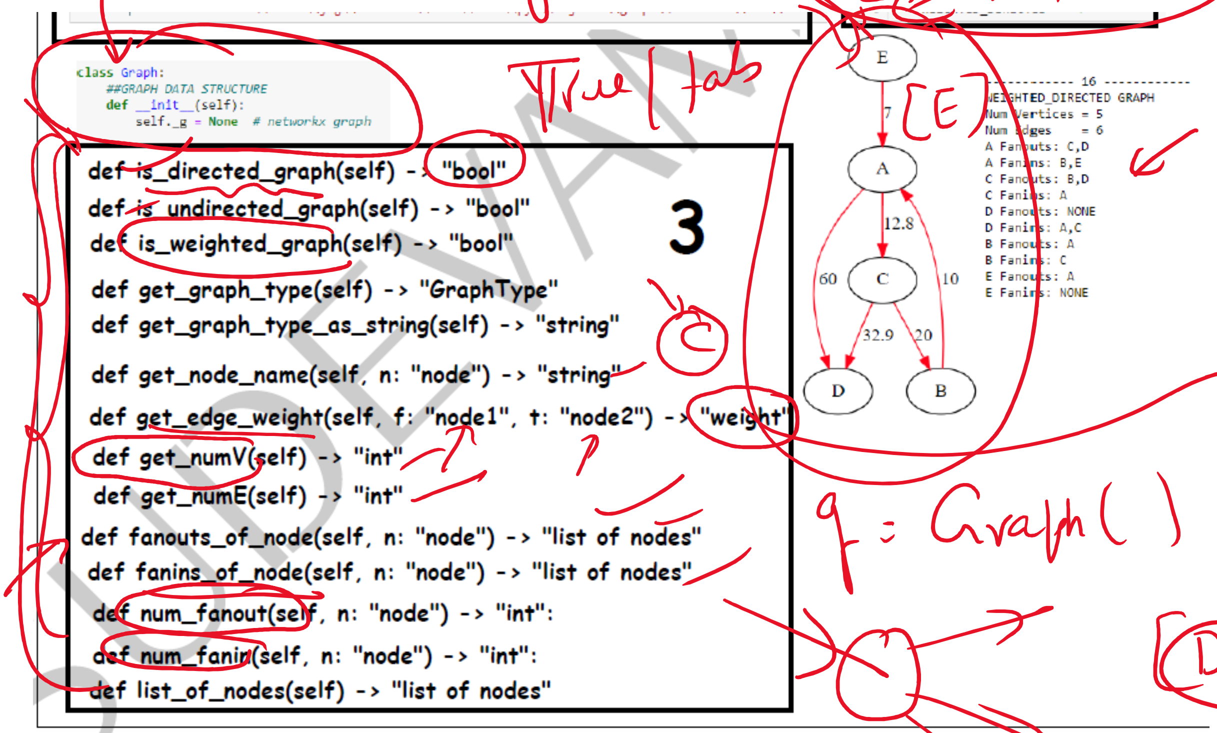
g. isdirected( )

Heap

h.addheap

24

Wednesday, November 8, 2023 7:04 PM



```python
class Graph:
    ##GRAPH DATA STRUCTURE
    def __init__(self):
        self._g = None  # networkx graph
```

```
def is_directed_graph(self) -> "bool"
def is_undirected_graph(self) -> "bool"
def is_weighted_graph(self) -> "bool"
def get_graph_type(self) -> "GraphType"
def get_graph_type_as_string(self) -> "string"
def get_node_name(self, n: "node") -> "string"
def get_edge_weight(self, f: "node1", t: "node2") -> "weight"
def get_numV(self) -> "int"
def get_numE(self) -> "int"
def fanouts_of_node(self, n: "node") -> "list of nodes"
def fanins_of_node(self, n: "node") -> "list of nodes"
def num_fanout(self, n: "node") -> "int":
def num_fanin(self, n: "node") -> "int":
def list_of_nodes(self) -> "list of nodes"
```

3

```
------------- 16 ------------
WEIGHTED_DIRECTED GRAPH
Num Vertices = 5
Num Edges    = 6
A Fanouts: C,D
A Fanins: B,E
C Fanouts: B,D
C Fanins: A
D Fanouts: NONE
D Fanins: A,C
B Fanouts: A
B Fanins: C
E Fanouts: A
E Fanins: NONE
```

Figure 18.18: Graph public functions

g

True / false

E, A, C, D, B

A

[E]

C

A, C

double

g = Graph( )

D, B, C

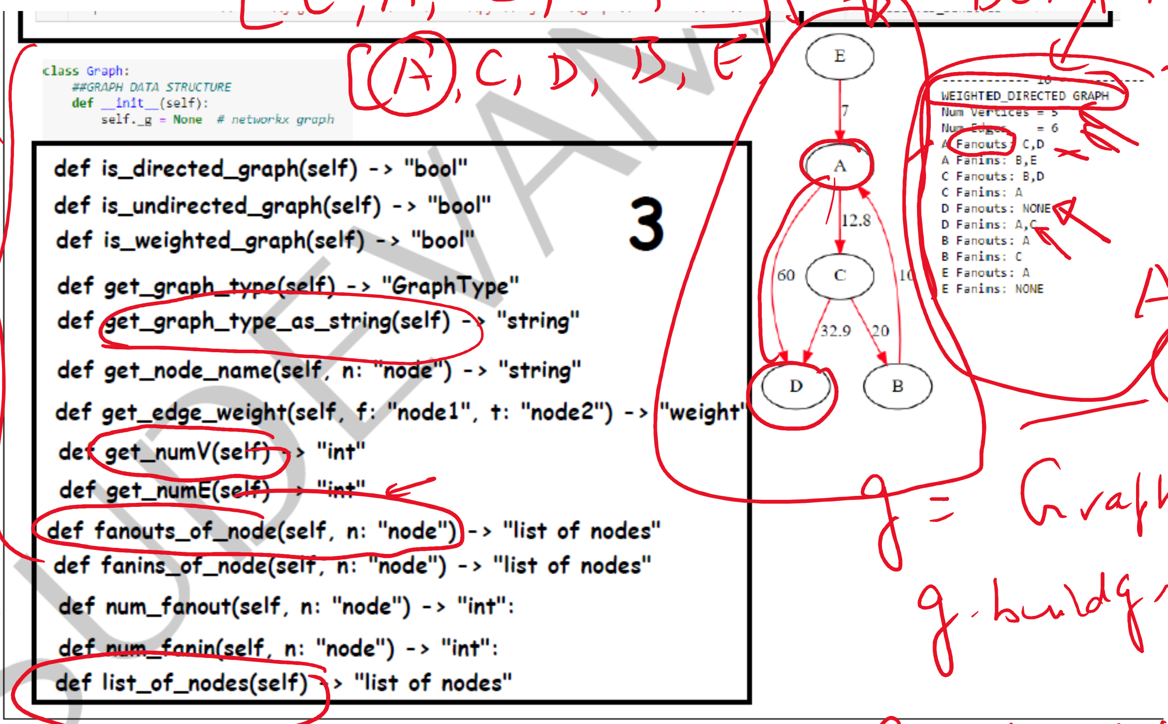
DS

[E, A, C, D, B]

DUMP File

[(A), C, D, B, E]

C, D



```
class Graph:
    ##GRAPH DATA STRUCTURE
    def __init__(self):
        self._g = None  # networkx graph
```

```
def is_directed_graph(self) -> "bool"
def is_undirected_graph(self) -> "bool"
def is_weighted_graph(self) -> "bool"
def get_graph_type(self) -> "GraphType"
def get_graph_type_as_string(self) -> "string"
def get_node_name(self, n: "node") -> "string"
def get_edge_weight(self, f: "node1", t: "node2") -> "weight"
def get_numV(self) -> "int"
def get_numE(self) -> "int"
def fanouts_of_node(self, n: "node") -> "list of nodes"
def fanins_of_node(self, n: "node") -> "list of nodes"
def num_fanout(self, n: "node") -> "int":
def num_fanin(self, n: "node") -> "int":
def list_of_nodes(self) -> "list of nodes"
```

3

```
------------ 10 ------------
WEIGHTED_DIRECTED GRAPH
Num Vertices = 5
Num Edges    = 6
A Fanouts: C,D
A Fanins: B,E
C Fanouts: B,D
C Fanins: A
D Fanouts: NONE
D Fanins: A,C
B Fanouts: A
B Fanins: C
E Fanouts: A
E Fanins: NONE
```

A

del.fil

g = Graph( )

g.buildgraph( )

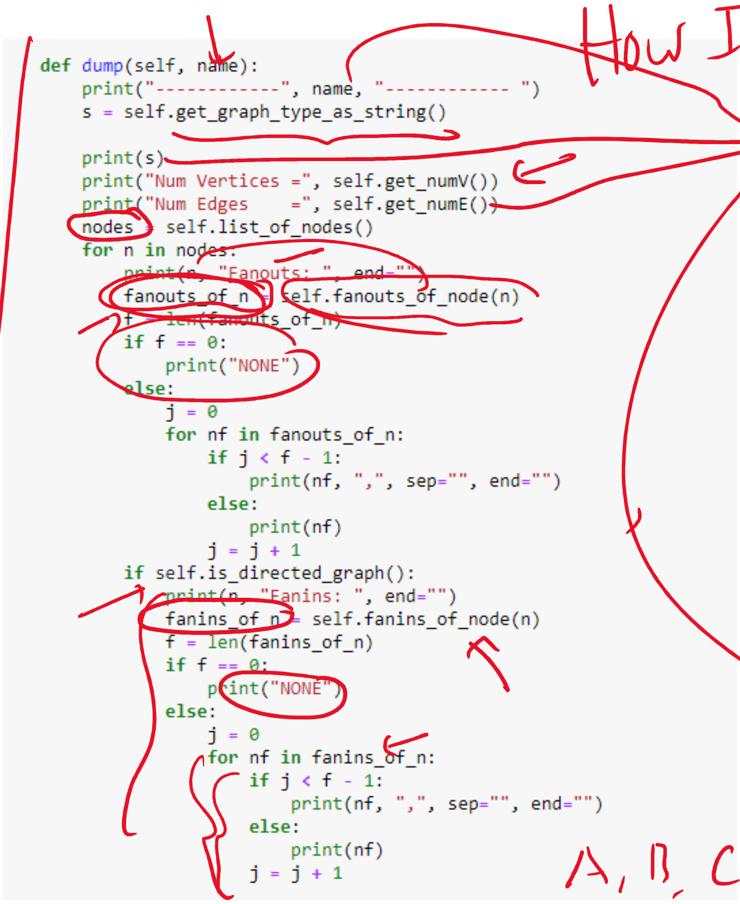
g.get_numV( )

Figure 18.18: Graph public functions

26

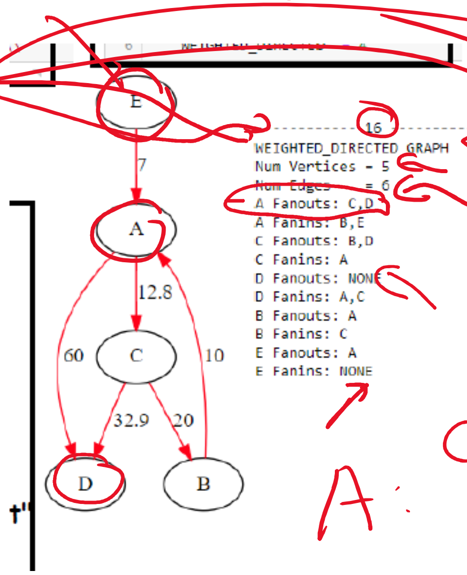
Wednesday, November 8, 2023 7:15 PM

How I g (g.dump("16")



```python
def dump(self, name):
    print("------------", name, "------------ ")
    s = self.get_graph_type_as_string()

    print(s)
    print("Num Vertices =", self.get_numV())
    print("Num Edges    =", self.get_numE())
    nodes = self.list_of_nodes()
    for n in nodes:
        print(n, "Fanouts: ", end="")
        fanouts_of_n = self.fanouts_of_node(n)
        f = len(fanouts_of_n)
        if f == 0:
            print("NONE")
        else:
            j = 0
            for nf in fanouts_of_n:
                if j < f - 1:
                    print(nf, ",", sep="", end="")
                else:
                    print(nf)
                j = j + 1
        if self.is_directed_graph():
            print(n, "Fanins: ", end="")
            fanins_of_n = self.fanins_of_node(n)
            f = len(fanins_of_n)
            if f == 0:
                print("NONE")
            else:
                j = 0
                for nf in fanins_of_n:
                    if j < f - 1:
                        print(nf, ",", sep="", end="")
                    else:
                        print(nf)
                    j = j + 1
```



```
------------ 16 ------------
WEIGHTED_DIRECTED GRAPH
Num Vertices - 5
Num Edges    = 6
A Fanouts: C,D
A Fanins: B,E
C Fanouts: B,D
C Fanins: A
D Fanouts: NONE
D Fanins: A,C
B Fanouts: A
B Fanins: C
E Fanouts: A
E Fanins: NONE
```

C, D, E,

A :

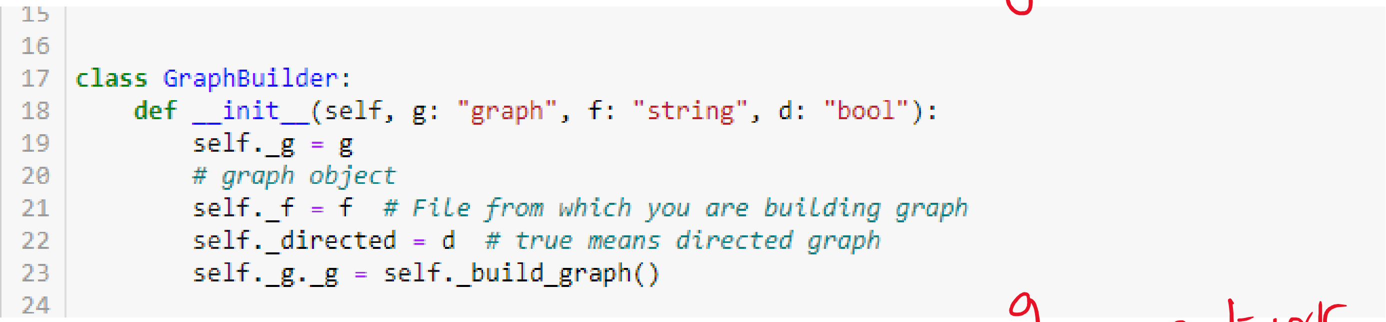
A

A, B, C

27

Wednesday, November 8, 2023 7:19 PM

g.



```
class GraphBuilder:
    def __init__(self, g: "graph", f: "string", d: "bool"):
        self._g = g
        # graph object
        self._f = f  # File from which you are building graph
        self._directed = d  # true means directed graph
        self._g._g = self._build_graph()

```

Undirect

true

13.txt

g

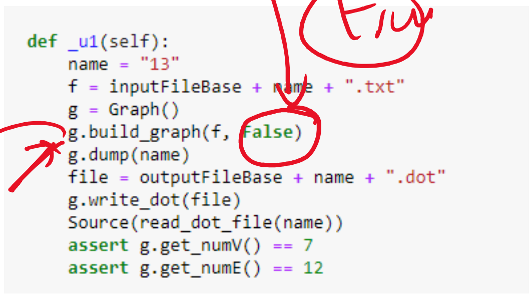
nextwork.

1 2
4 5
1 7

1 2
4 5
1 7

F

1 2
4 5
1 7



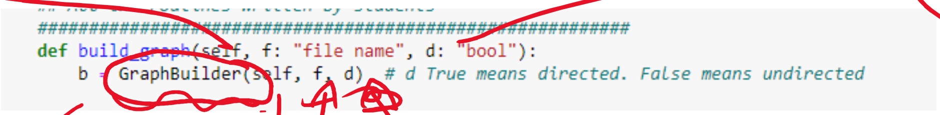
```
def _u1(self):
    name = "13"
    f = inputFileBase + name + ".txt"
    g = Graph()
    g.build_graph(f, False)
    g.dump(name)
    file = outputFileBase + name + ".dot"
    g.write_dot(file)
    Source(read_dot_file(name))
    assert g.get_numV() == 7
    assert g.get_numE() == 12
```

13.txt.

d = true

false

Graph3D

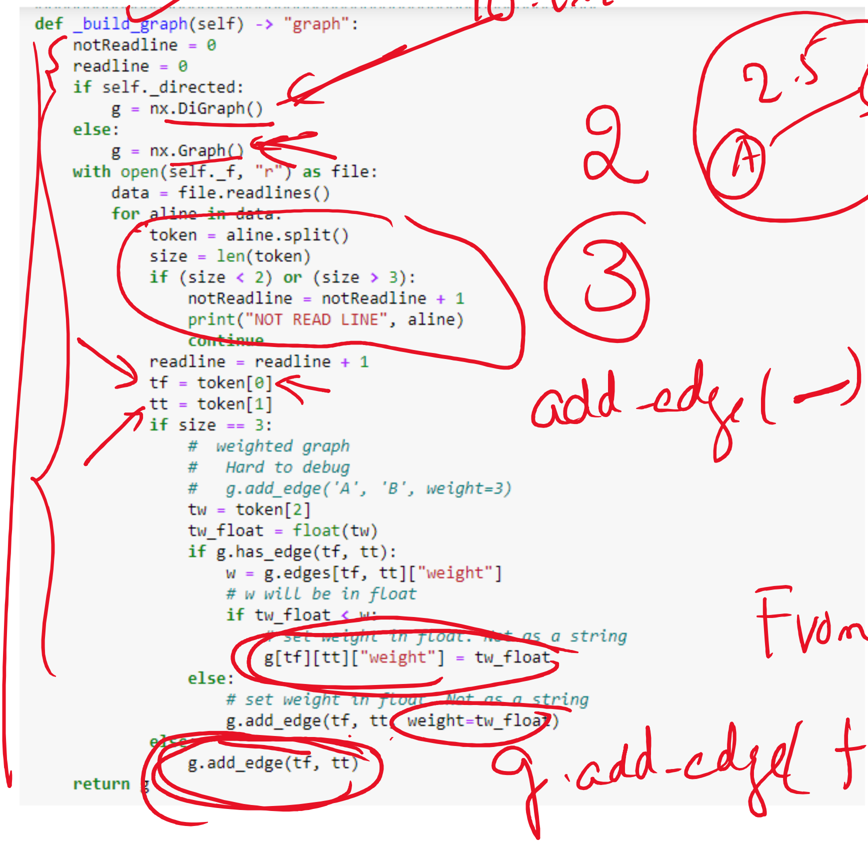


```
## Add all routines written by students
##############################################################
def build_graph(self, f: "file name", d: "bool"):
    b = GraphBuilder(self, f, d)  # d True means directed. False means undirected
```

28

Wednesday, November 8, 2023 7:26 PM

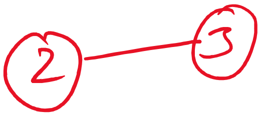
10.txt



```python
def _build_graph(self) -> "graph":
    notReadline = 0
    readline = 0
    if self._directed:
        g = nx.DiGraph()
    else:
        g = nx.Graph()
    with open(self._f, "r") as file:
        data = file.readlines()
        for aline in data:
            token = aline.split()
            size = len(token)
            if (size < 2) or (size > 3):
                notReadline = notReadline + 1
                print("NOT READ LINE", aline)
                continue
            readline = readline + 1
            tf = token[0]
            tt = token[1]
            if size == 3:
                #  weighted graph
                #   Hard to debug
                #   g.add_edge('A', 'B', weight=3)
                tw = token[2]
                tw_float = float(tw)
                if g.has_edge(tf, tt):
                    w = g.edges[tf, tt]["weight"]
                    # w will be in float
                    if tw_float < w:
                        # set weight in float. Not as a string
                        g[tf][tt]["weight"] = tw_float
                else:
                    # set weight in float. Not as a string
                    g.add_edge(tf, tt, weight=tw_float)
            else:
                g.add_edge(tf, tt)
    return g
```

2.5 B A

1 2
3 5
4 7



g.

2

3

add_edge(—)

#

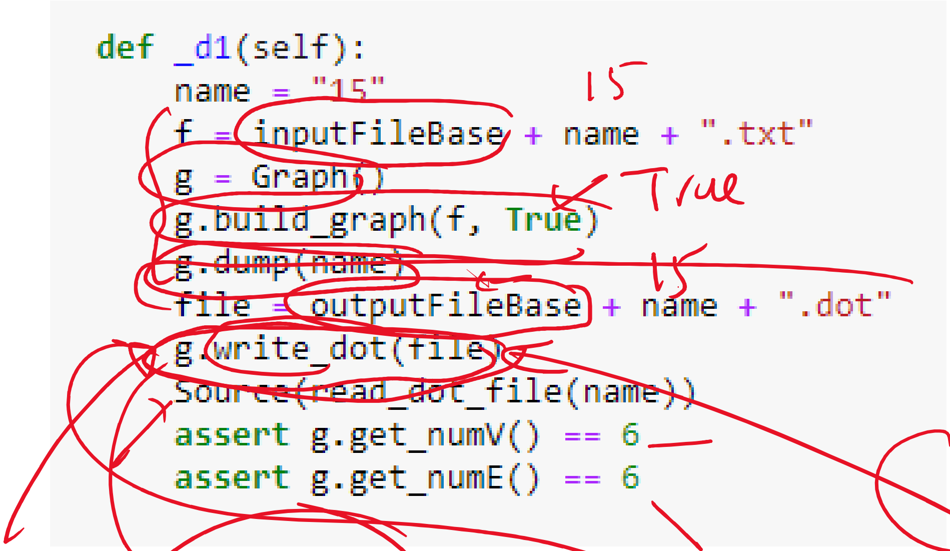
1 2 2.5
2 3 4.0

From City

W

g.add_edge(from, to) g

Wednesday, November 8, 2023 7:34 PM



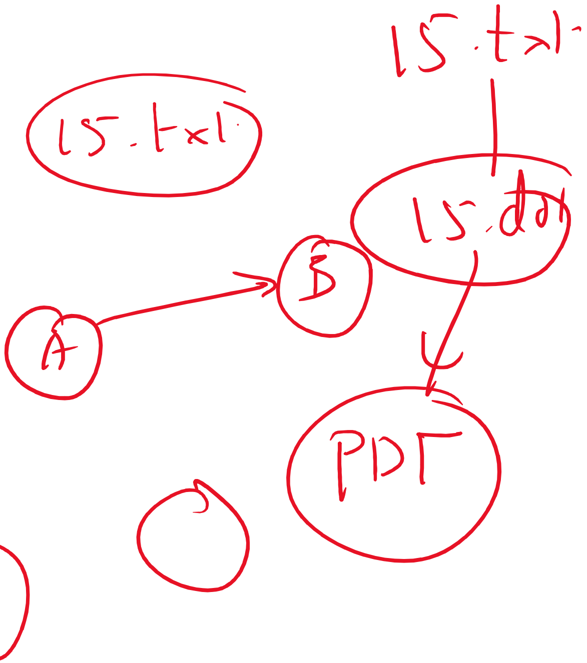
```
def _d1(self):
    name = "15"
    f = inputFileBase + name + ".txt"
    g = Graph()
    g.build_graph(f, True)
    g.dump(name)
    file = outputFileBase + name + ".dot"
    g.write_dot(file)
    Source(read_dot_file(name))
    assert g.get_numV() == 6
    assert g.get_numE() == 6
```

15

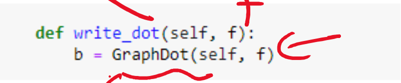
True

15

Graph



f



```
def write_dot(self, f):
    b = GraphDot(self, f)
```

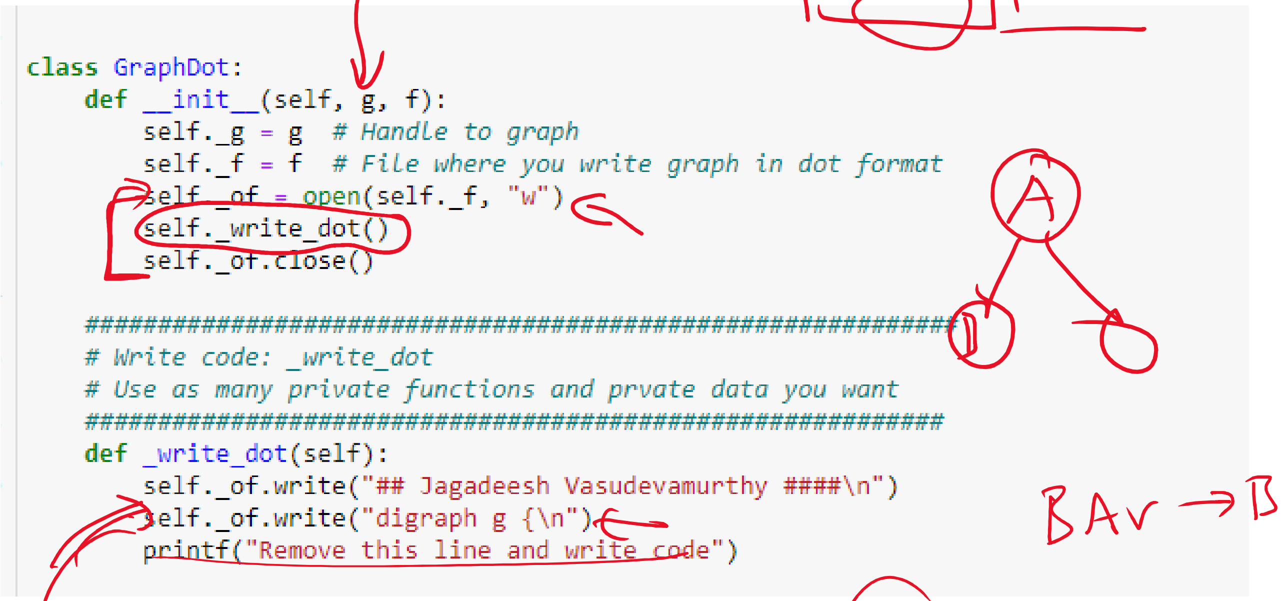
GraphDot (Sel g, f

30

Wednesday, November 8, 2023 7:37 PM

Wednesday, November 8, 2023 7:40 PM

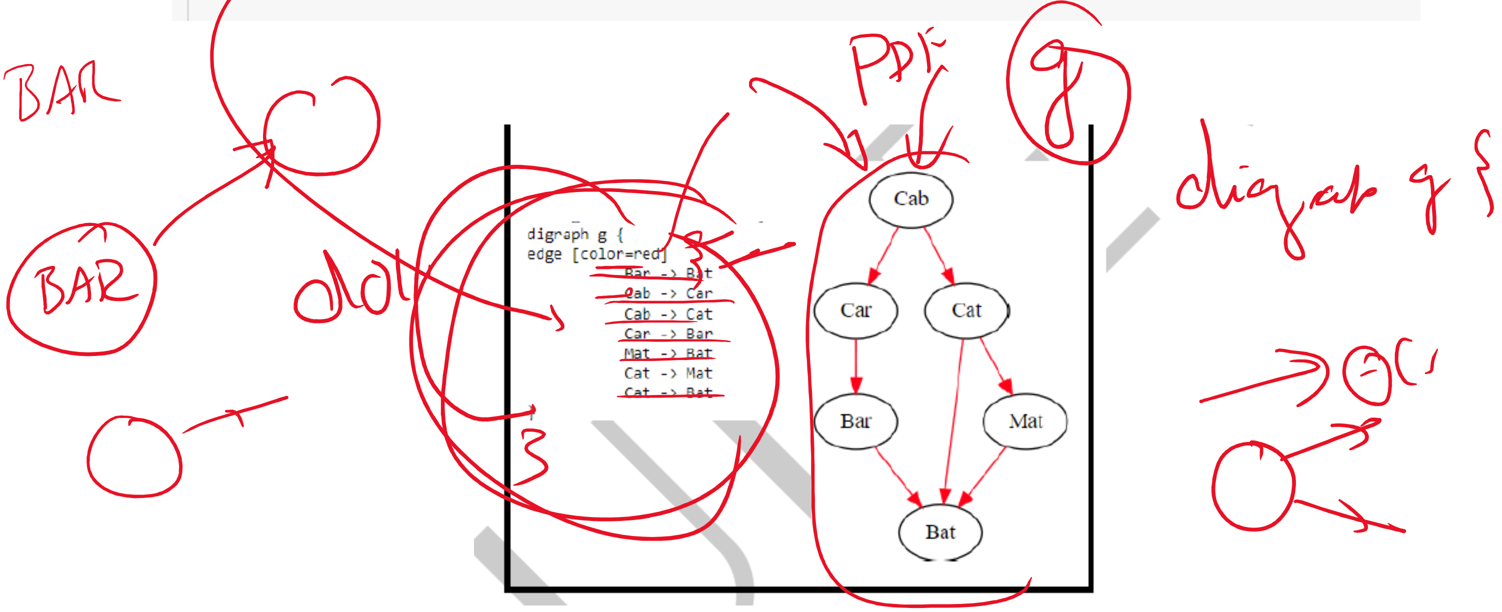
network x
Public



```
class GraphDot:
    def __init__(self, g, f):
        self._g = g  # Handle to graph
        self._f = f  # File where you write graph in dot format
        self._of = open(self._f, "w")
        self._write_dot()
        self._of.close()

    #################################################################
    # Write code: _write_dot
    # Use as many private functions and prvate data you want
    #################################################################
    def _write_dot(self):
        self._of.write("## Jagadeesh Vasudevamurthy ####\n")
        self._of.write("digraph g {\n")
        printf("Remove this line and write code")
```

BAr -> BA



```
digraph g {
edge [color=red]
          Bar -> Bat
          Cab -> Car
          Cab -> Cat
          Car -> Bar
          Mat -> Bat
          Cat -> Mat
          Cat -> Bat
}
```