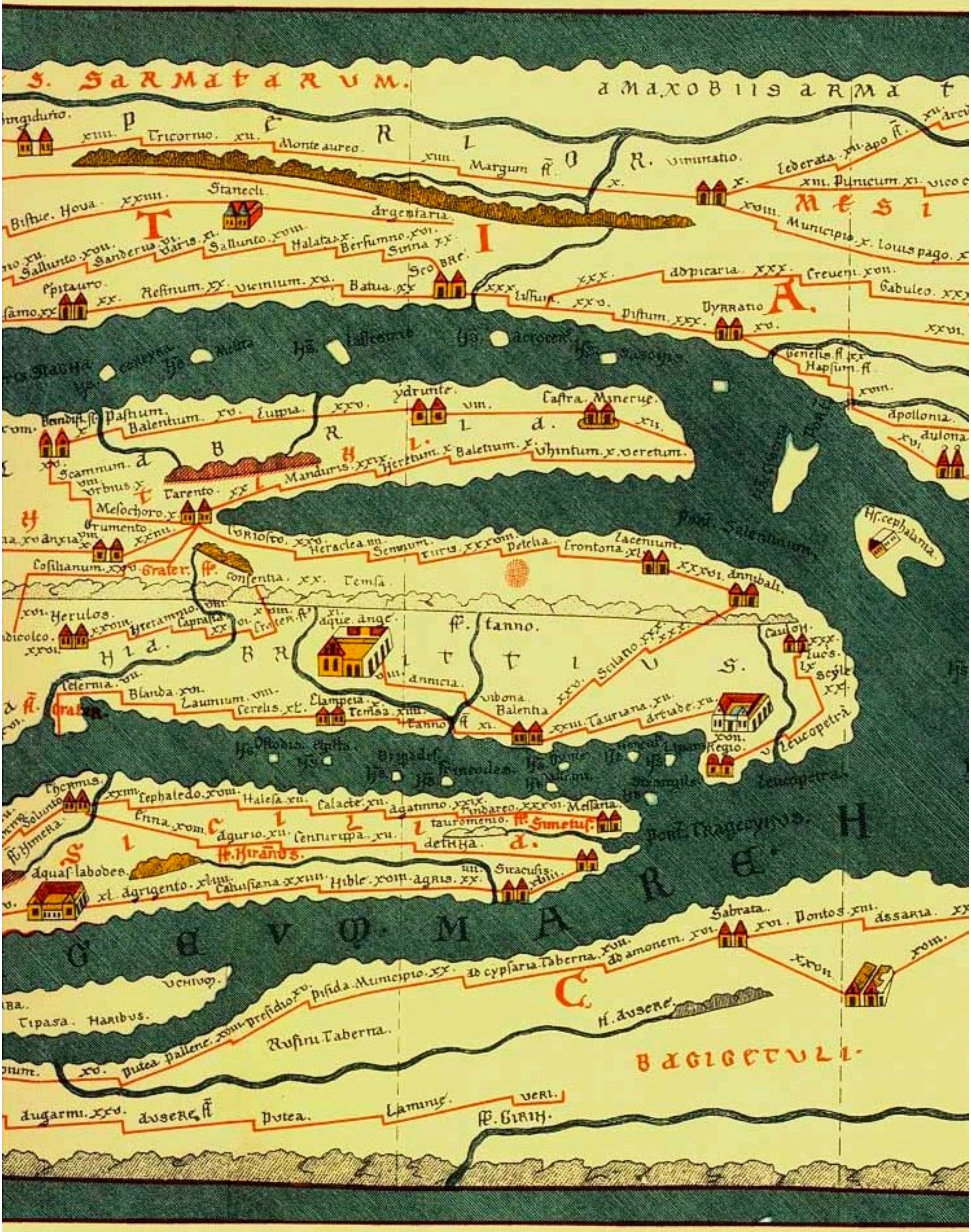
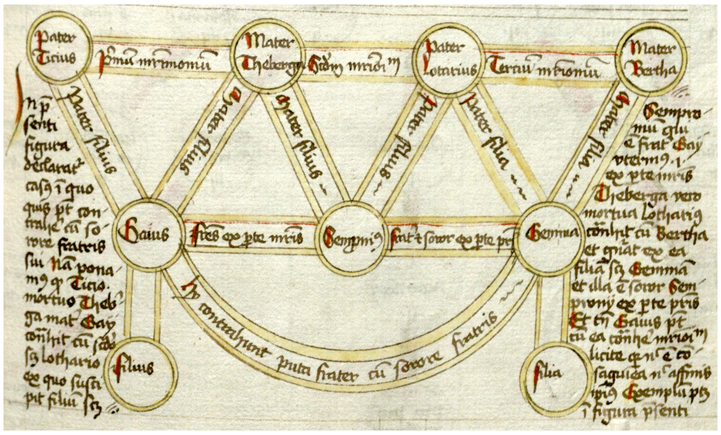
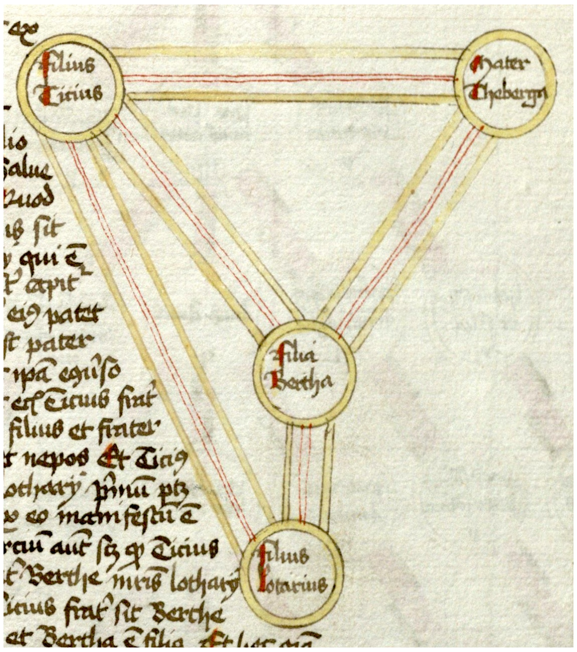
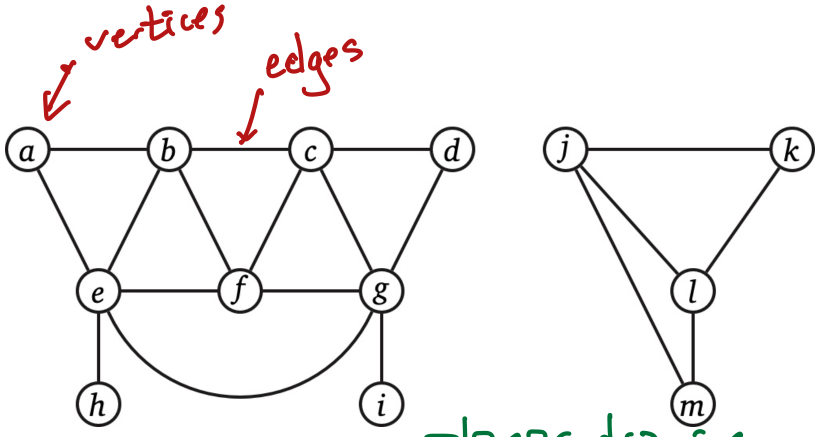

S. SARMATARUM.
AMAXOBIIS ARMAT
Tricornio. xii.
Monte aureo.
Margum
R. Viminatio.
Lederata. xvi. apo
xiii. Punicum. xi. vico
MESI
Municipio. x. Lovis pago.
Stanecli
Bistue. Nova. xxiiii.
argentaria
T
I
A.
Sallunto. xvi.
Sanderua. vi.
Saris. xi.
Sallunto. xviii.
Halata. x.
Berfumno. xvi.
Sinna. xx.
Sco BRE
Epitauro
Resinum. xx.
Vicinium. xvi.
Batua. xx.
Lissum
adpicaria. xxx.
Creueni. xvi.
Gabuleo.
Dyrratio
Pistum. xxx.
Beneventum
Hapsum
Apollonia.
Aulona
Brundisium
Pastium.
Balentium. xv.
Lupia.
Ydrunte.
Castra Minerue.
Scammum.
Orbius.
Melochoro.
Tarento.
Manduris. xxix.
Heretum.
Baletium.
Uhintum. x. Veretum.
Grumento.
Cosilianum. xxv.
Crater. fl.
Consentia. xx.
Temsa.
Turiosto.
Heraclea.
Semnum.
Turis.
Petelia.
Crontona.
Lacenium.
Annibali.
Hs. Cephalania.
Port. Salentinum.
Herulos.
Caprasia.
Crater fl.
aque ange
fl. tanno.
BRITTIUS.
Scilatio. xxx.
Caulon
Lucos.
Scyle
Leucopetra
Cerernia.
Blanda. xvi.
Laumium.
Cerelis.
Clampeia.
Temsa.
Tanno
Annicia.
Vibona Balentia
Tauriana.
Arciade.
Regio.
Leucopetra.
Hs. Vulcani.
Strongile.
Port. Tragecynus.
Thermis.
Solunto
Himera
Cephaledo. xviii.
Halesa.
Calacte.
Agatinno.
Tindareo.
Messana.
Enna.
Agurio.
Centuripa.
Tauromenio.
fl. Simetus.
Ethna.
SICILIA.
fl. Hirandis.
aquas labodes.
Agrigento.
Calvisiana.
Hible.
Agris.
Siracusis.
H
MARE
G
R
GEVUM MARE
Sabrata.
Pontos.
Assaria.
Ad amonem.
Ad cypsaria Taberna.
Pisida Municipio.
Presidio.
Putea Pallene.
Rusini Taberna.
C.
fl. Ausere.
BAGIGETULI.
Tipasa.
Haribus.
Dugarmi.
Ausere fl.
Putea.
Lamnus.
Veri.
fl. Girin.


Pater Ticius
Mater Theberga
Pater Lotarius
Mater Bertha
Gaius
Gemma
Filius
Filia


Filius Ticius
Mater Theberga
Filia Bertha
Filius Lotarius



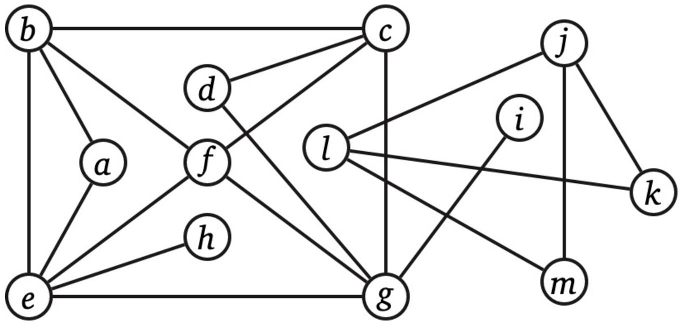
planar drawing

V — vertices
any finite
nonempty set

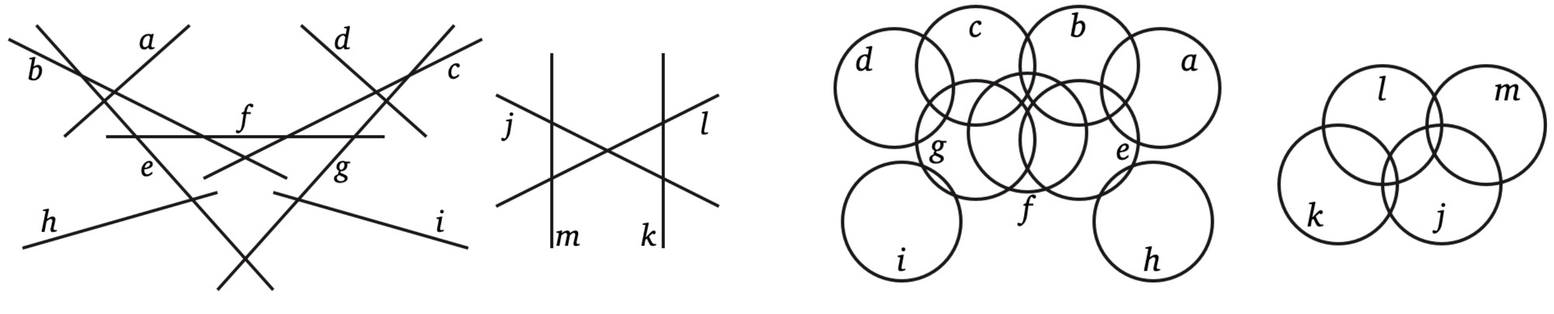
E — edges
= pairs of vertices

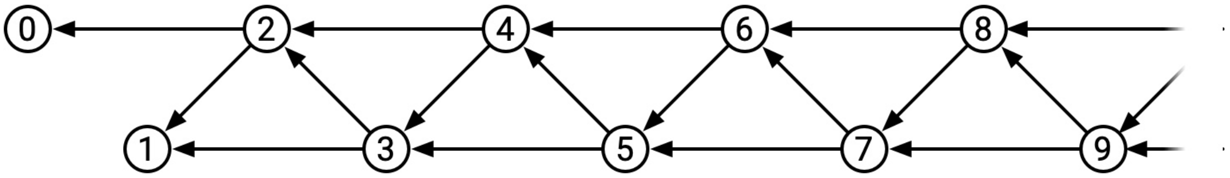
undirected: $\{u,v\} = uv$
$= vu$

directed: $(u,v) = u \to v$



intersection graphs

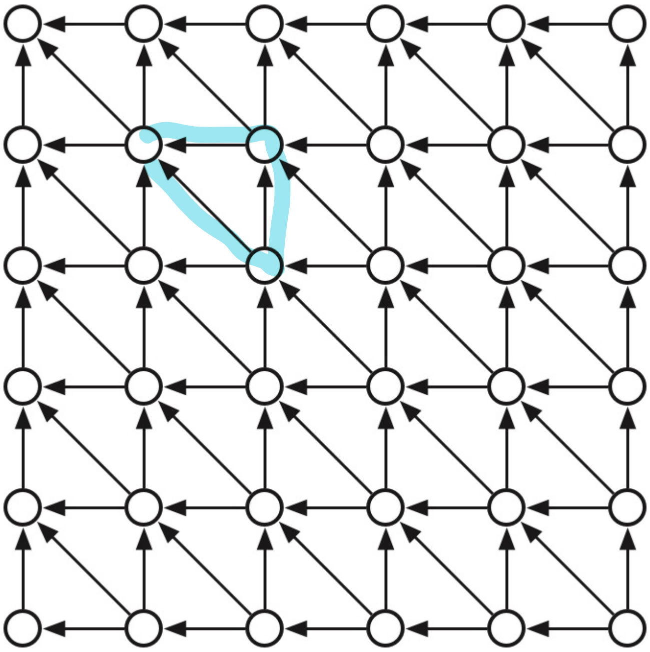


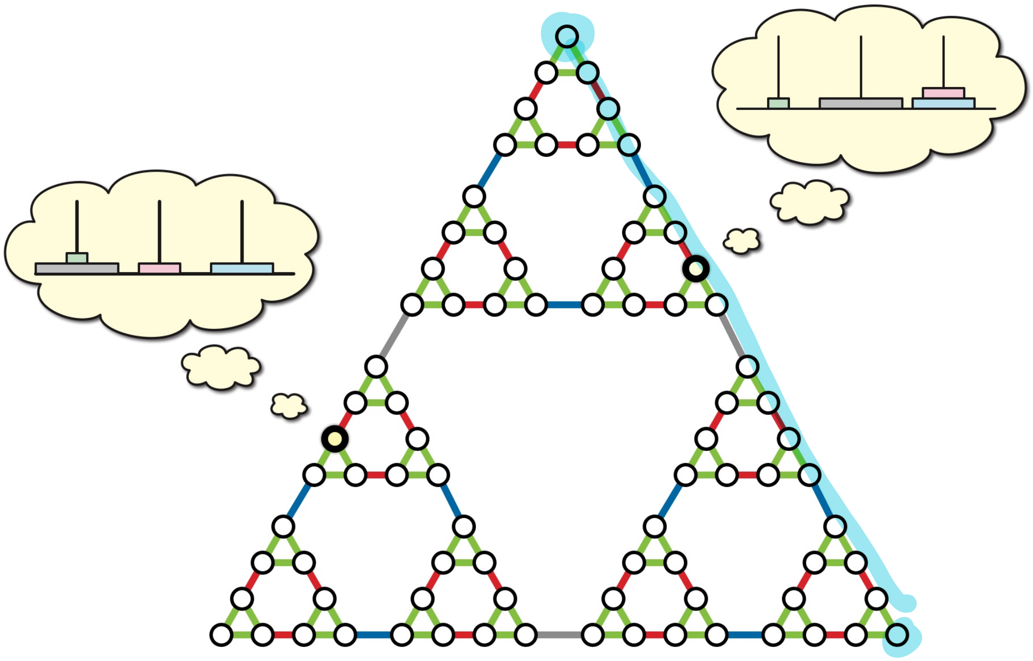


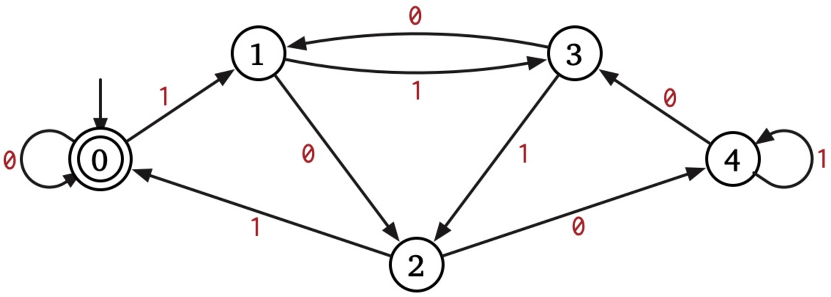
$F_n = F_{n-1} + F_{n-2}$

Edit distance $\Longrightarrow$

Directed acyclic graphs





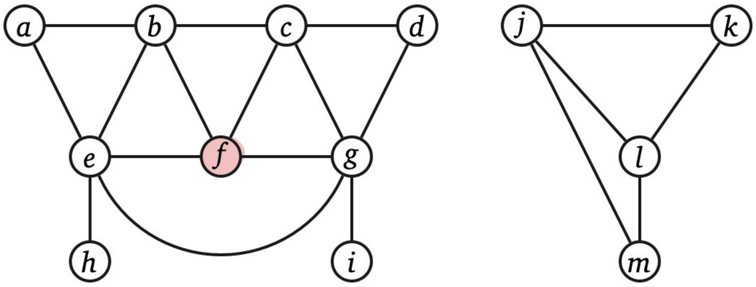


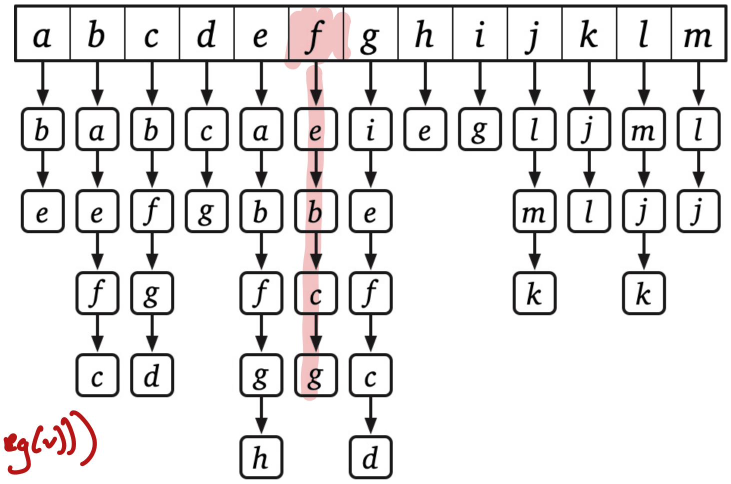
Data structures

v.color   color[v]

v.mark   mark[v]

DEFAULT:



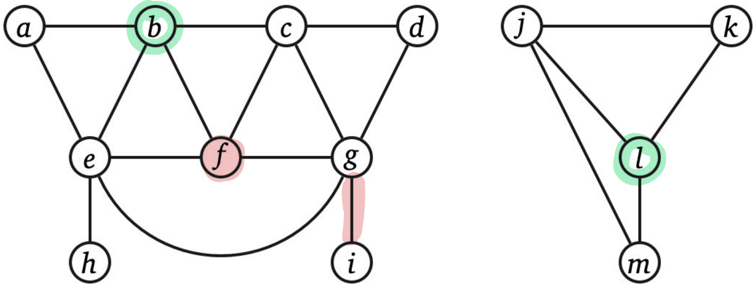


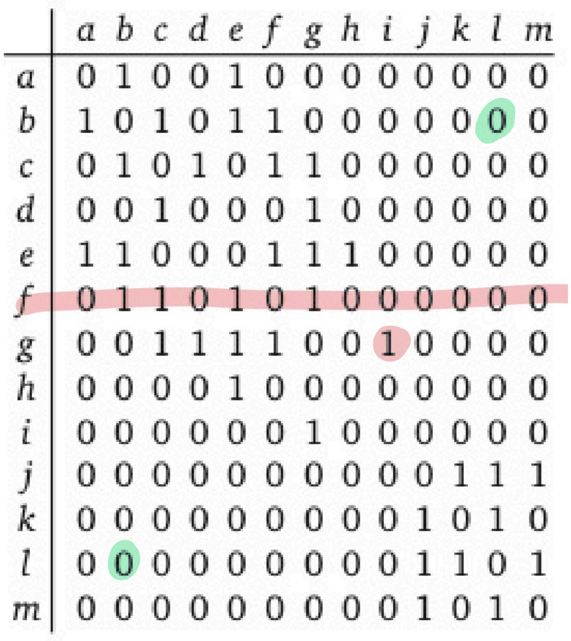
adjacency list

check if uv∈E $O(\min(\deg(u),\deg(v)))$

list all neighbors of v in $O(\deg(v))$

$O(V+E)$ space





	a	b	c	d	e	f	g	h	i	j	k	l	m
a	0	1	0	0	1	0	0	0	0	0	0	0	0
b	1	0	1	0	1	1	0	0	0	0	0	0	0
c	0	1	0	1	0	1	1	0	0	0	0	0	0
d	0	0	1	0	0	0	1	0	0	0	0	0	0
e	1	1	0	0	0	1	1	1	0	0	0	0	0
f	0	1	1	0	1	0	1	0	0	0	0	0	0
g	0	0	1	1	1	1	0	0	1	0	0	0	0
h	0	0	0	0	1	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	1	0	0	0	0	0	0
j	0	0	0	0	0	0	0	0	0	0	1	1	1
k	0	0	0	0	0	0	0	0	0	1	0	1	0
l	0	0	0	0	0	0	0	0	0	1	1	0	1
m	0	0	0	0	0	0	0	0	0	1	0	1	0

Adjacency matrix:

uv∈E? $O(1)$ time

list all neighbors of v $O(V)$ time

$O(V^2)$ space

Depth-First search

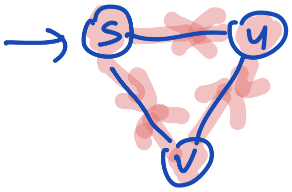
```
RECURSIVEDFS(v):
    if v is unmarked
        mark v
        for each edge vw
            RECURSIVEDFS(w)
```

```
ITERATIVE~~D~~BFS(s):
    PUSH(s)
    while the ~~stack~~ queue is not empty
        v ← ~~Pop~~ PULL
        if v is unmarked
            mark v
            for each edge vw
                PUSH(w)
```

```
WHATEVERFIRSTSEARCH(s):
    put s into the bag ← 1
    while the bag is not empty
        take v from the bag ⇐ O(E) times
        if v is unmarked
            mark v ⇐ O(V) times
            for each edge vw
                put w into the bag ← ** O(E) times
```

O(V+E)

insert
remove one
empty?



BAG: ~~s~~ ~~u~~ ~~v~~ u ~~s~~ s v

```
WHATEVERFIRSTSEARCH(s):
    put (∅, s) in bag
    while the bag is not empty
        take (p, v) from the bag            (⋆)
        if v is unmarked
            mark v
            parent(v) ← p
            for each edge vw                (†)
                put (v, w) into the bag     (⋆⋆)
```

```
WFSAll(G):
    for all vertices v
        unmark v
    for all vertices v
        if v is unmarked
            WhateverFirstSearch(v)
```

```
CountComponents(G):
    count ← 0
    for all vertices v
        unmark v
    for all vertices v
        if v is unmarked
            count ← count + 1
            WhateverFirstSearch(v)
    return count
```

$O(V+E)$ time

```
CountAndLabel(G):
    count ← 0
    for all vertices v
        unmark v
    for all vertices v
        if v is unmarked
            count ← count + 1
            LabelOne(v, count)
    return count
```

```
⟨⟨Label one component⟩⟩
LabelOne(v, count):
    while the bag is not empty
        take v from the bag
        if v is unmarked
            mark v
            comp(v) ← count
            for each edge vw
                put w into the bag
```