Dynamic programming

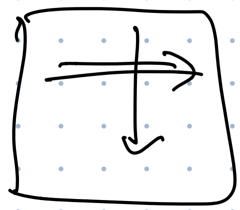
- English description

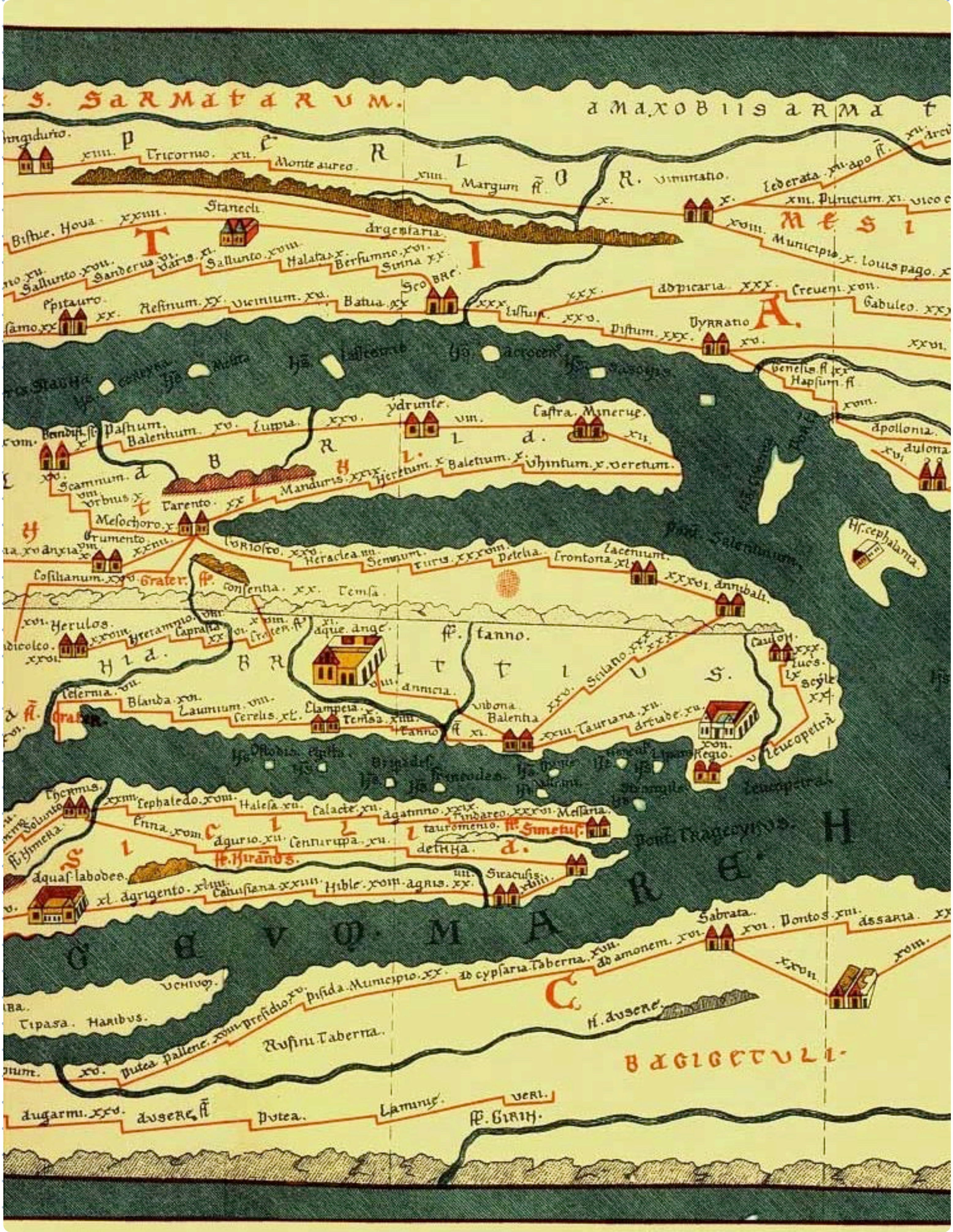
  "Edit(i,j) is edit distance from A[1..i] to B[1..j]"

- Greedy. optimization

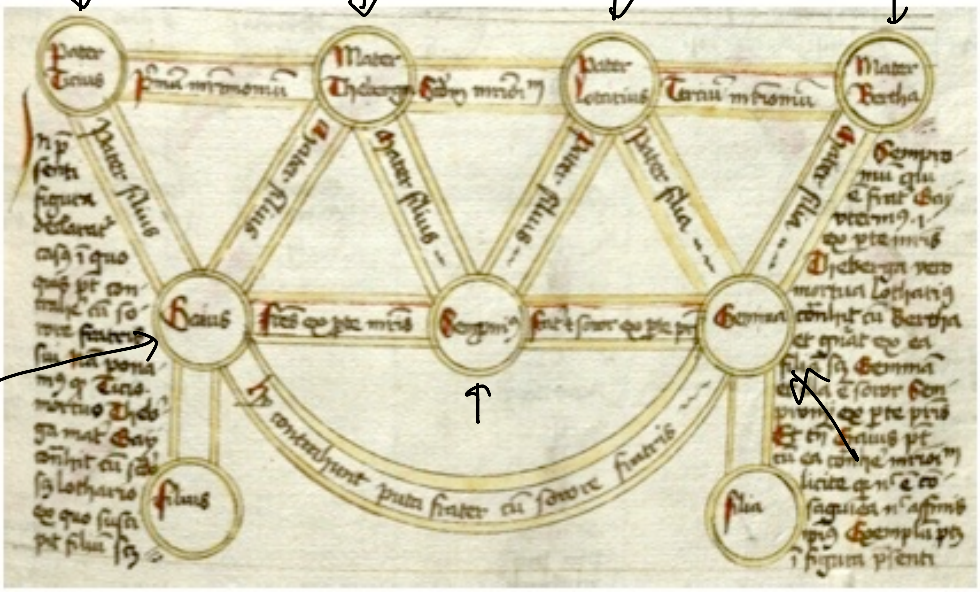
$$\text{Edit}(i,j) = \begin{cases} \min\begin{Bmatrix} 1 + \text{Edit}(i-1,j) \\ 1 + \text{Edit}(i,j-1) \\ 1 + \text{Edit}(i-1,j-1) \end{Bmatrix} & \text{if } i \neq j \\ \min\begin{Bmatrix} \cancel{1 + \text{Edit}(i-1,j)} \\ \cancel{1 + \text{Edit}(i,j-1)} \\ \text{Edit}(i-1,j-1) \end{Bmatrix} & \text{if } i = j \end{cases}$$

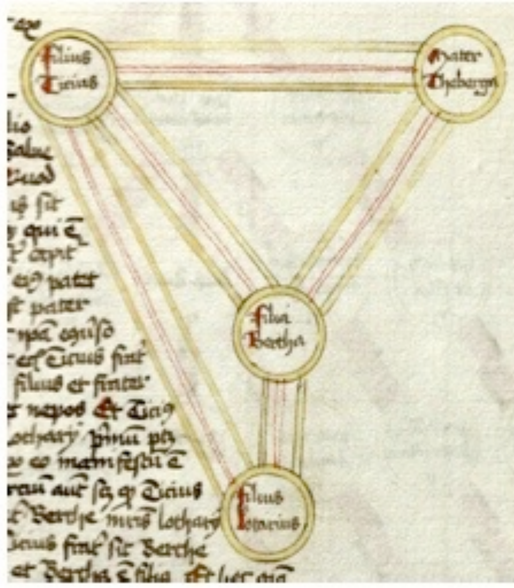
- Memo vs. iterative

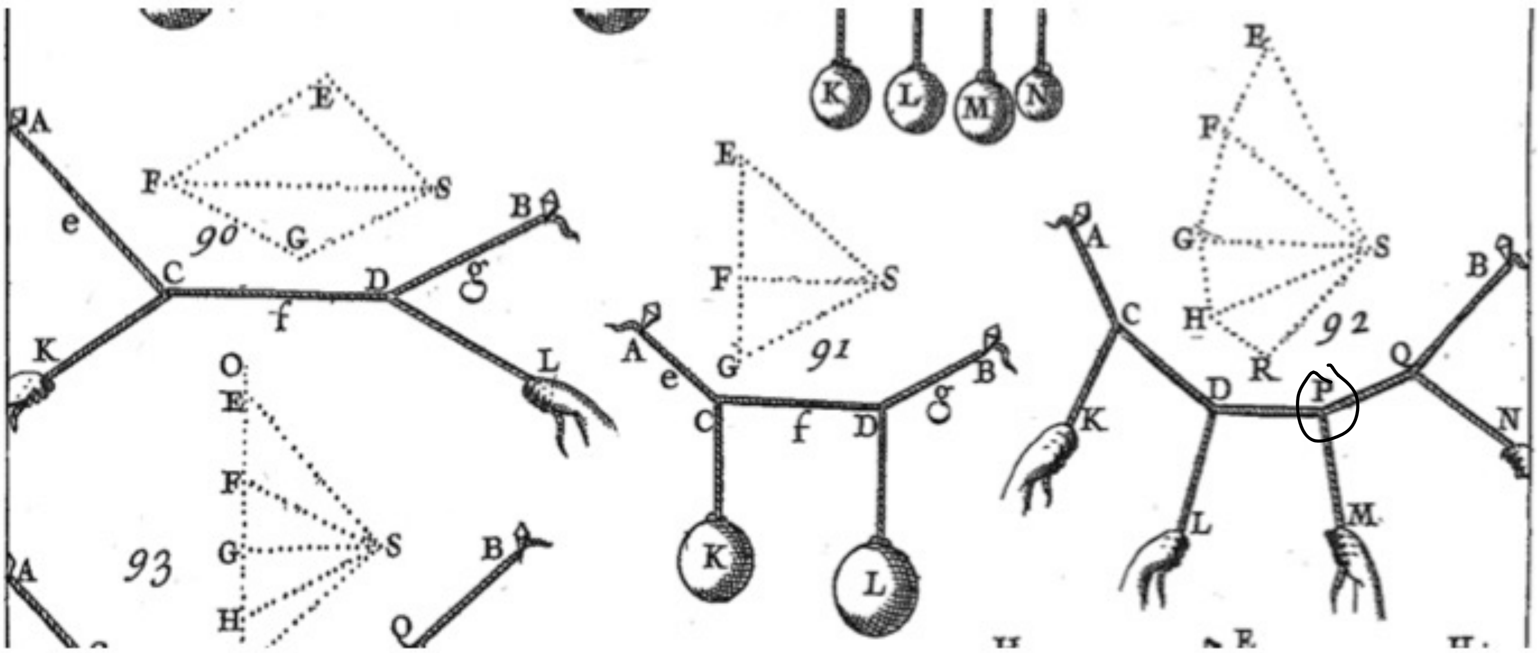




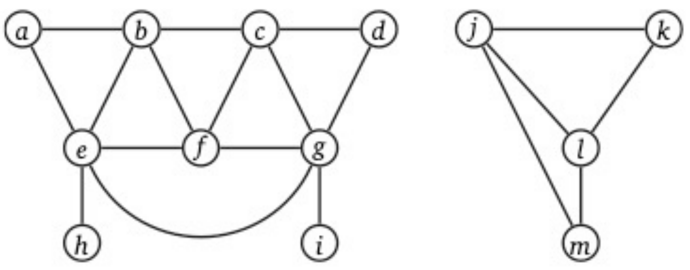
S. SARMATARUM.
AMAXOBII SARMAT
Tricornio
Monte aureo
Margum
Viminatio
Lederata
Punicum
Stanecli
Argentaria
MESI
Bersumno
Sinna
Batua
Scodra
Aesinum
Vicinium
Epitauro
Adipicaria
Creueni
Gabuleo
Dyrratio
Pistum
Apollonia
Aulona
Ydrunte
Castra Minerue
Balentium
Lupia
Pasthum
Baletium
Uhintum
Ueretum
Manduris
Tarento
Mesochoro
Grumento
Anxia
Heraclea
Petelia
Crontona
Lacenium
Annibali
Cosilianum
Crater
Consentia
Temsa
Aque ange
Tanno
Scilatio
Caulon
Blanda
Laumium
Cerelis
Clampeia
Vibona Balentia
Tauriana
Decastadium
Regio
Leucopetra
Cephaledo
Halesa
Calacte
Agatinno
Tindareo
Messana
Thermis
Solunto
Enna
Agurio
Centuripa
Tauromenio
Fl. Simetus
Aquas labodes
Agrigento
Calvisiana
Hible
Agris
Siracusis
Fl. Hirandus
SICILIA
SALENTINUM
MARE
Sabrata
Pontos
Assaria
Pisida Municipio
Cypsaria Taberna
Ad amonem
Rusfini Taberna
Tipasa
Haribus
Putea Pallene
Presidio
Fl. Ausere
BAGIGETULI
Ausere fl.
Putea
Lamnie
Veri
Fl. Girin
Dugarmi

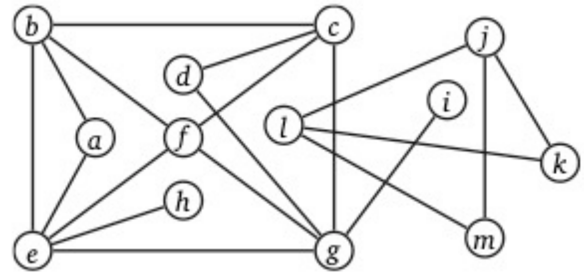

Pater Titius
Mater Theberga
Pater Lotarius
Mater Bertha
Gaius
Gemma
filius
filia


Filius Titius
Mater Theberga
filia Bertha
filius Lotarius


90
91
92
93

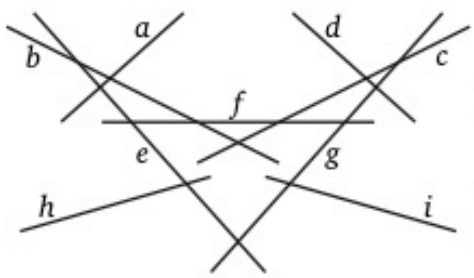
13 vertices $\{a, b, \ldots, m\}$ #edges $= \{ab, ae, be, ch, \ldots\}$

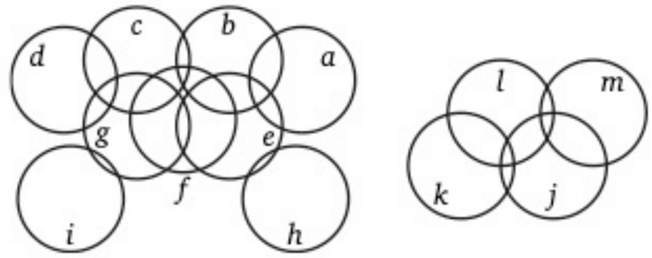




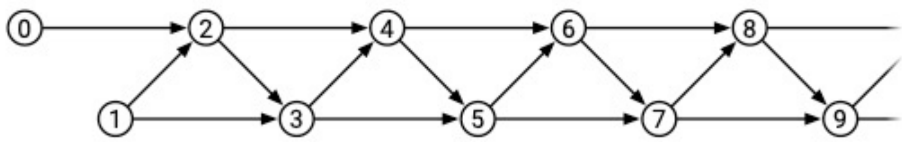
sing: ~~vertex~~   ~~vertice~~   node

pl. ~~vertices~~   node.

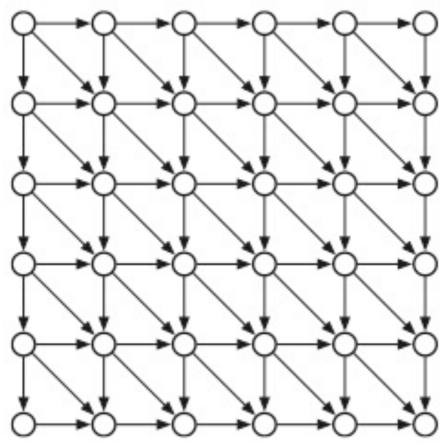


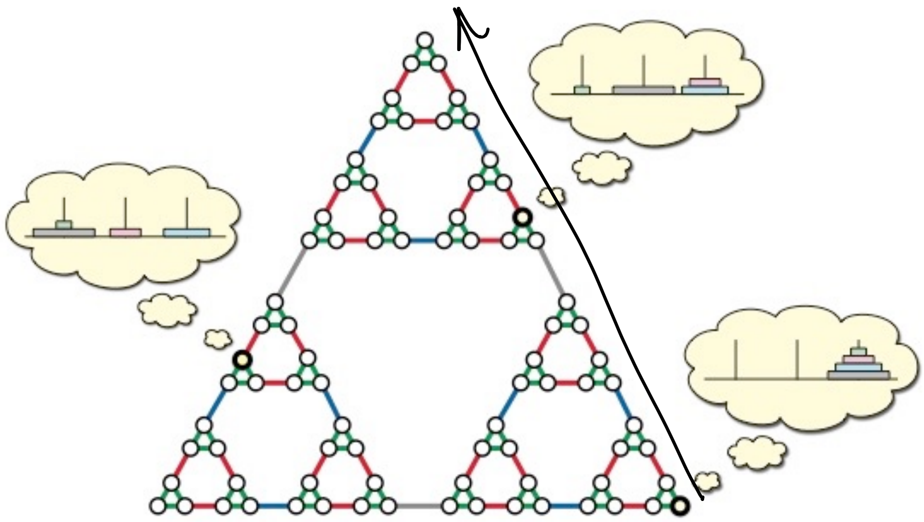


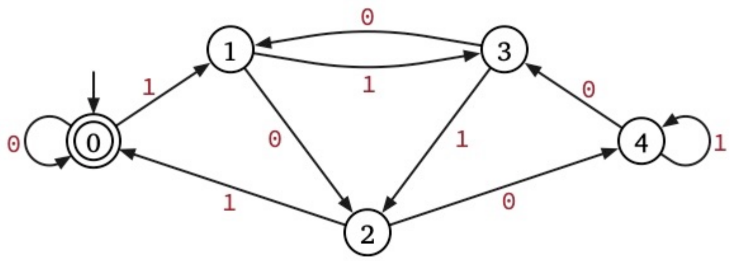
$$F_n = \begin{cases} 0 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ F_{n-1} + F_{n-2} & \text{otherwise,} \end{cases}$$



$$Edit(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min \left\{ \begin{array}{c} Edit(i-1,j)+1 \\ Edit(i,j-1)+1 \\ Edit(i-1,j-1) + [A[i] \neq B[j]] \end{array} \right\} & \text{otherwise} \end{cases}$$







product construction

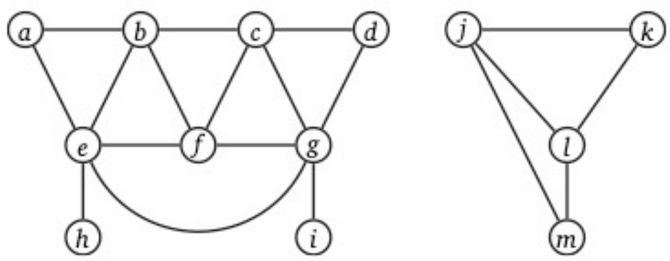
subset construction

Graph is a pair of sets (V, E)

V = vertices (any nonempty finite set)

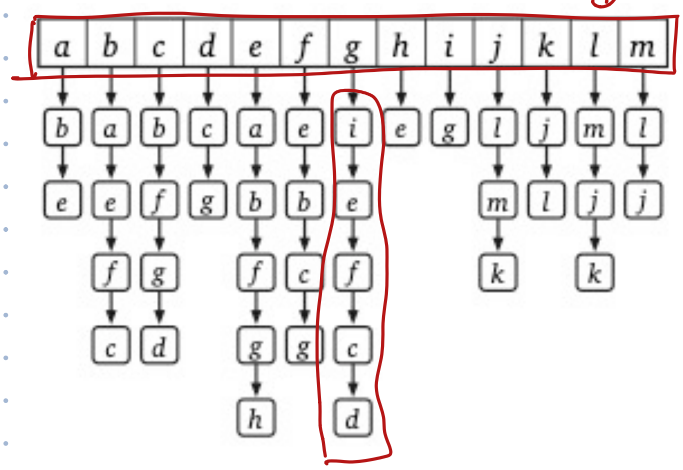
E = edges $\left\{\begin{matrix}\text{ordered}\\ \text{unordered}\end{matrix}\right\}$ pairs of vertices

Data Structures



Adjacency List

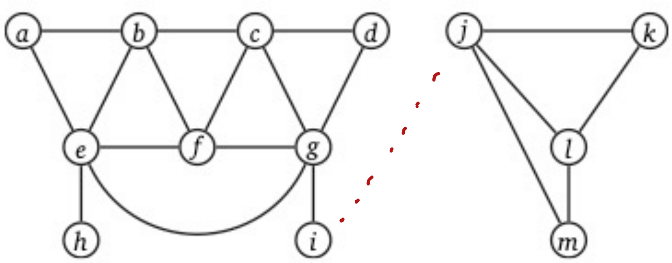
indexed by nodes



incident edges
adjacent vertices

Space : $O(V+E)$

Adjacency matrix



	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

Space : $\Theta(V^2)$

2d array Adj[1..V, 1..V]

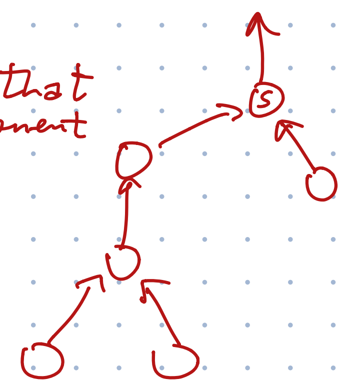
```
RecursiveDFS(v):
    if v is unmarked
        mark v
        for each edge vw
            RecursiveDFS(w)
```

```
IterativeDFS(s):
    Push(s)
    while the stack is not empty
        v ← Pop
        if v is unmarked
            mark v
            for each edge vw
                Push(w)
```

```
WhateverFirstSearch(s):
    put s into the bag
    while the bag is not empty
        take v from the bag
        if v is unmarked
            mark v
            for each edge vw
                put w into the bag
```

① Every vertex connected to s is marked and nothing else

② parent edges define a spanning tree of that component




```
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   (⋆⋆)
```

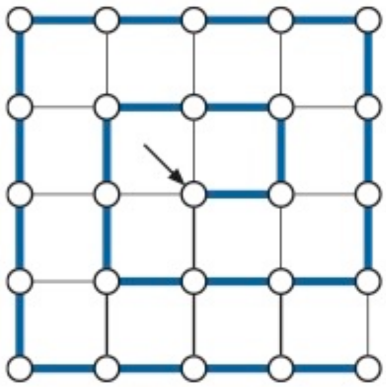
O(E)

O(V)

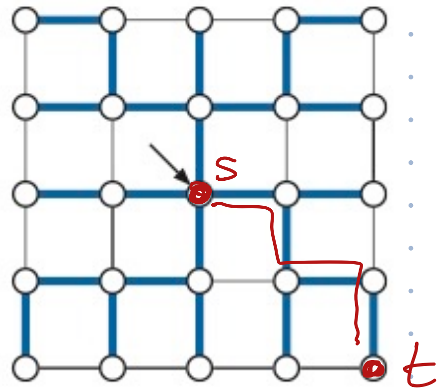
O(E)

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

depth (stack)



breadth (queue)



shortest paths

```
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
```

```
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
```