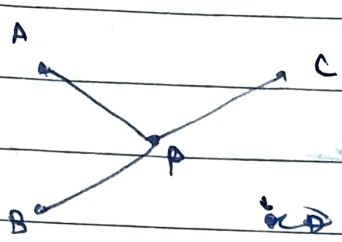
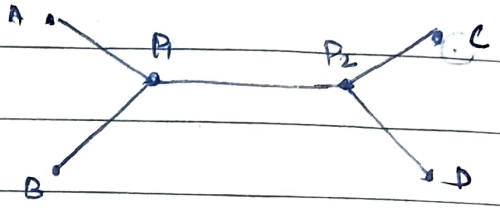
Page No.
Date

GEOMETRIC ALGO

STEINER TREE PROBLEM



3-node steiner



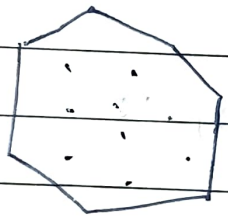
4-node steiner.

CONVEX HULL

(sorting of geometry)

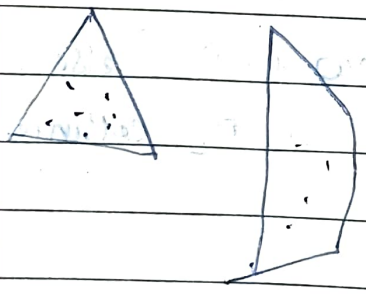
3 algorithms.

$\Theta(n \cdot h)$ $\Theta(n^2)$ $\Theta(n \log n)$



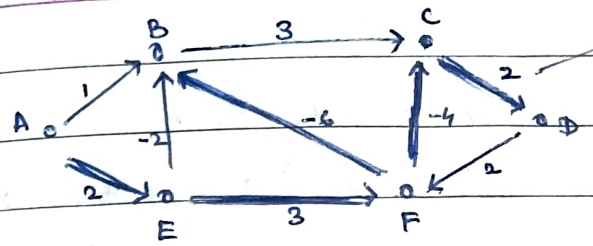
no. of pts. on convex hull (jarvis algo)

ronald-graham scanning algo



- VLSI layout
- CV bounding box (polygon)
- PR pattern recognition.
  delaunay triangulation.
  voronoi diagram
- LP linear programming.
  simplex (boundary pts.).

EX SHORTEST PATH vs MST



① go on forever

node	dist	parent
A	0	nil
B	∞ ~~10~~ -1	~~A~~ ~~E~~ F
C	∞ ~~4~~ ~~3~~ 1	~~D~~ ~~B~~ F
D	∞ ~~6~~ 3	~~C~~ C
E	∞ 2	A
F	∞ 5	E

2 cycles. BCDFB 3 2 2 -6 ①

FCDF -4 2 2 ②

~~B D C D F B C E E B A~~

queue empty

algorithm stops.

bellman ford algo (e.n) updating forever?

BFS scanning runs for (n-1) phases.

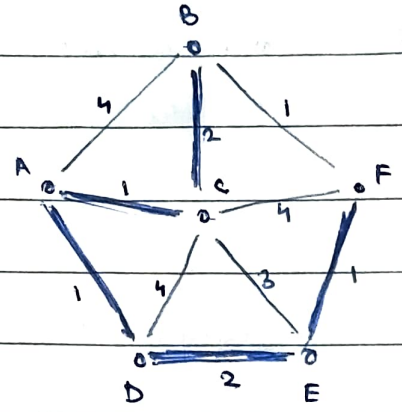
gyjan's arg.

① all the paths with 0 degrees (edges)

② all the paths with 1 edge have their distances correct.

③ all the paths with 2 edge dist. to source have their distances correct.

④ ...



scan (v)
for n adj. to v
if (dist (v) + weight (n, v)) <
(dist (u)) then
dist (u) = dist (v) + weight (v, u)
parent of (u) = v.

	node	dist	parent
Source →	A	0	nil.
	B	∞ 13	C
	C	∞ 1	A
	D	∞ 1	A
	E	∞ 13	D
	F	∞ 14	E

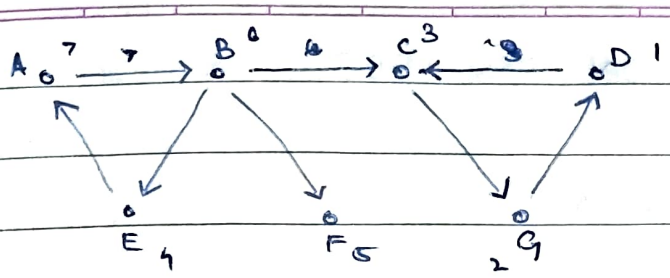
③ general case → [bellman-ford algo.]
(Tarjar DS & N/w algo)
(breadth-first scanning algo).

COMPLEXITY

adj. list
heap
O(elog n)

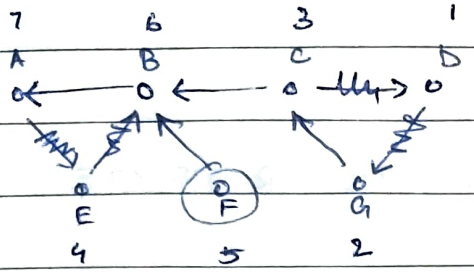
adj. matrix
$\Theta(n^2)$

all pairs shortest (floyd-warshall)
$\Theta(n \cdot e \log n)$ / $\Theta(n \cdot n^2)$
$\Theta(n^3)$



adjacency matrix reverse

edge → transpose



SHORTEST PATH

① weighted undirected graph (positive sets).

dijkstra's $\Theta(e \log n)$ / $O(n^2)$

(adj. list + heap) (adj. matrix).

② DAG (+, −, wts) no cycles $\Theta(e)$

③ general graph: (+, −) no neg. wt. cycles $\Theta(e \cdot n)$

④ dijkstra's algo.

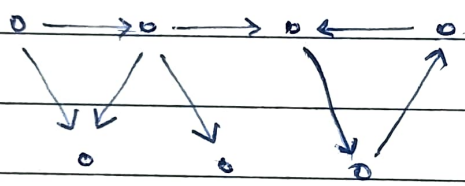
post order traversal OF DFS TREE

D C E B F H G A

1 2 3 4 5 6 7 8

← topological sort

STRONGLY CONNECTED COMPONENTS FOR DIRECTED GRAPH.

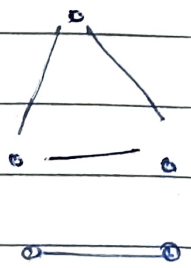


algo for SCC

O(e) ① DFS find the finishing times for each node.

O(e) ② reverse the edges in the graph.

O(e) ③ run DFS on the reversed graph in reverse order of finishing time.



O(e+n)

↓

O(e).

Page No. _____
Date _____

AMORTIZED ANALYSIS

stack

push: $\Theta(1) \longrightarrow \Theta(n)$

pop : $\Theta(1)$

variation: double the space allocated. Copy (no. elements)

space alloc:	push	copy (no.)
2	2nd elem.	1
4	3rd elem.	2
8	5th elem	4
16	9th elem	8
32	17th elem	16

$31 + 17 = 48$

$31 + 33$ G4 $= 3 \times 16$

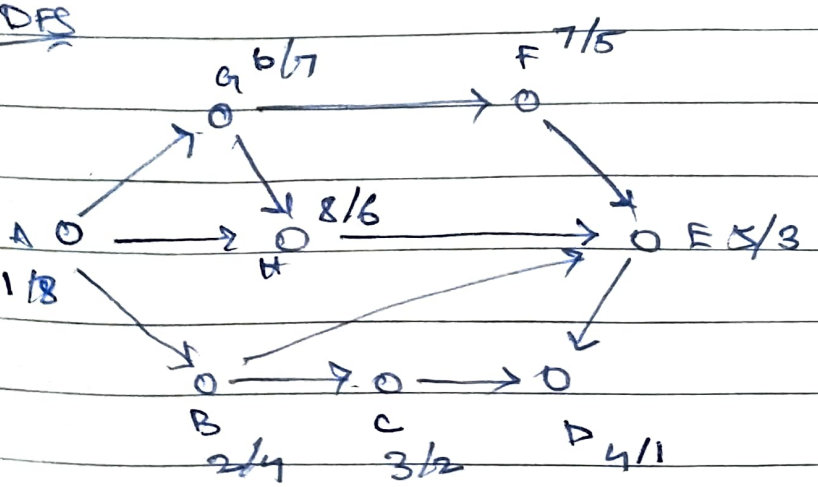
2×32 ∴

amortized analysis push $\Theta(3.n) \longrightarrow \Theta(2.n)$

A G H F B E C D

BFS (alphabetic order) QUEUE

DFS



DFS tree

topological sorting

directed graph

- tree edge
- cross edge

IMPROVEMENT 1

weighted union

- join smaller weight tree to bigger height tree

worst case for find $O(\log n)$

↓

$O(\alpha \log^* n)$

$\alpha \leq 5$ for practical purposes.

$2^{2^{2^{2^2}}} = 4$ amortized complexity

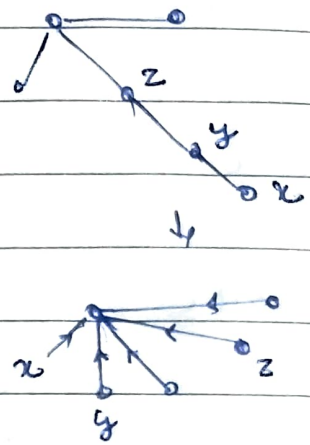
amortized analysis.

(with weighted union, with path compression) $\log^*$

BFS DFS.

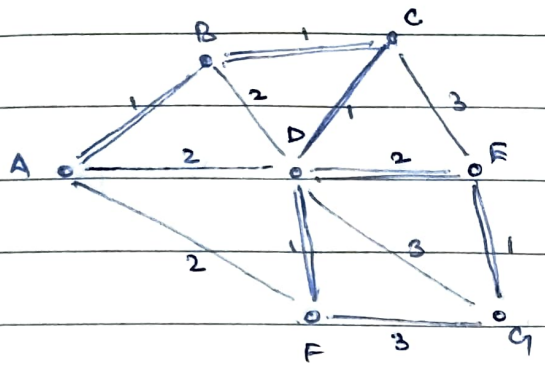
IMPROVEMENT 2

(path compression).



Page No. _____
Date _____

MST KRUSKAL'S ALGO

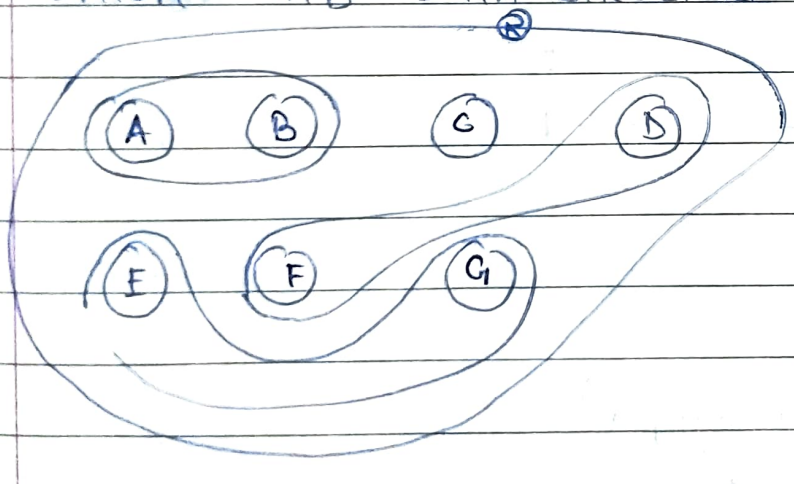


- sort the edges
  $O(e \log e)$
  $= O(e \log n)$

AB → EG → DF → CD → BC → DE

Complexity: $O(en)$.

UNION - FIND DATA STRUCTURE

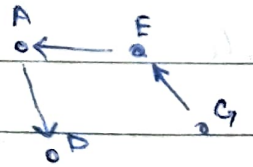


7 equivalent classes (singleton)

↓

1 equivalence class = MST

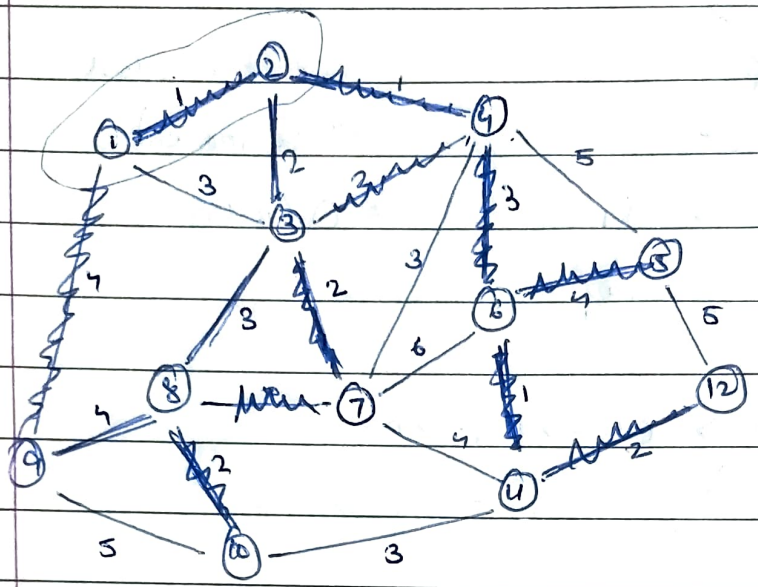
for each {x, y} in the sorted order



if (find(x) ≠ find(y)) then
union (find(x), find(y))
Store {x, y} in spanning tree

MST PRIM'S ALGO

order of nodes removal. edges constant



2 arrays.

parent	array indices	value	min heap
nil	1	0	
1	2	1	1

ANALYSIS

- building heap : O(n)
- delete from heap : O(n log n)
- reheap (changing value).
- adjust values of edges connected with node.
- return : O(log n)

PRIM'S MST

- O(elogn) ← sparse graph

using adj. matrix

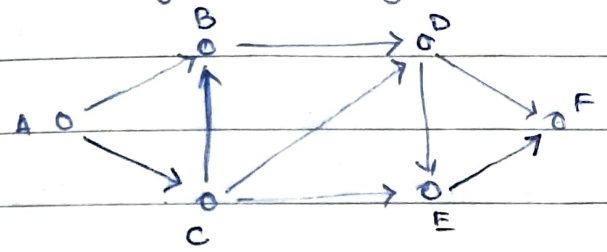
- $O(n^2)$ ← dense graph

Page No. _____
Date _____

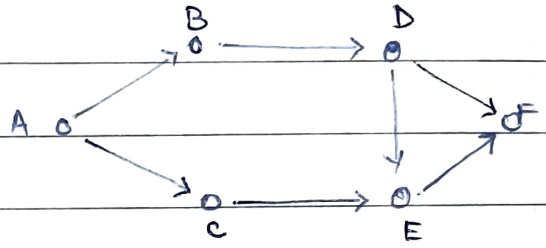
GRAPH ALGORITHMS

order of nodes removal — edges constant

ex topological sorting.

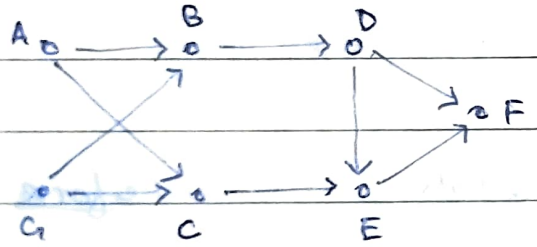


A C B D E F (topological sort)



~ A B D C E F

~ A C B D E F



A G B C D E F

adj. matrix (incidence) (dense graph)

adj. list (sparse graph).

until graph is not empty

- find all nodes with 0 indegree
- delete the node
- print or output that node

repeat.

O (e+n)

O (ne)

0 A → B → C → null

0 ~~1~~ ~~2~~ B → D → null

0 ~~1~~ ~~2~~ C → E → null

0 D → E → F → null

0 ~~1~~ ~~2~~ E → F → null

0 ~~1~~ ~~2~~ F → null

0 G → B → C → null

$O(e+n)$ ← eliminates.

Can be precomputed.

TRADE OFF

DS ← & algorithms.

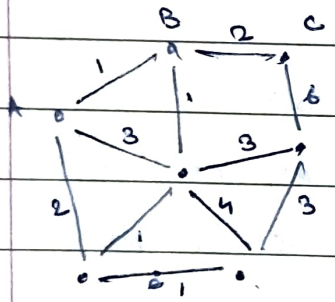
F	E	D	C	B	G	A

TS works on DAG ← $O(e)$

adj. matrix, (incidence) (dense graph)

adj. list (sparse graph).

ex MST



prim's partial tree n-1 edges.

kruskal's forest tree.

prim's	kruskal's
AB	AB
BF	BF
FG	FG
GE	GE
BC	BC
ED/FD	FD/ED

brute force

generate all possible subtree get least wt.

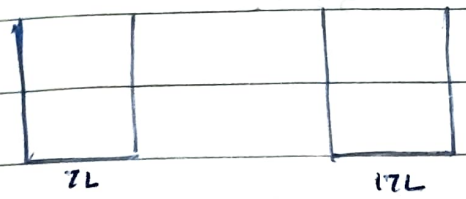
$O(4^n)$.

catalan #

WHY JUG OTHR FORMULA !

Page No. _____
Date _____

2 JUG PUZZLE

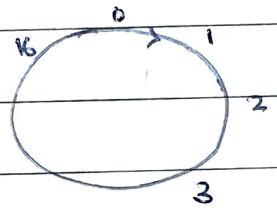


$1 = 5(7) - 2(17)$ 5 hops

$1 = 5(17) - 12(7)$ 12 hops.

	7L → 17L	17L → 7L
1	0	17
2	7	10
3	14	3
4	4	13
5	11	6
6	1	16
7	8	9
8	15	2
9	5	12
10	12	5
11	2	15
12	9	8
13	16	1
14	6	11
15	13	4
16	3	14
17	10	7

modules arithmetic



EUCLID's GCD ALGORITHM

$\gcd(17, 7) = \gcd(7, 3)$
$= \gcd(3, 1) = \gcd(1, 0)$
$=$

gcd 17 7
= gcd 7 3
= gcd 3 1
= gcd (1) 0

complexity approx. of the order of no. of digits of smaller no. (quite fast!)

app. of gcd algorithm in 2 jug problem.

gcd 17 7	$1 = 7 - 2(3)$
=gcd 7 3	$= 7 - 2\{17 - 2(7)\}$
gcd 3 1	$= 7 - 2(17) + 4(7)$
gcd (1) 0	$= 5(7) - 2(17)$

$1 = 5(17) - 12(7)$

ex 100, 46

gcd 100 46	$2 = ?(100) - ?(46)$
=gcd 46 8	$2 = 8 - 6$
=gcd 8 6	$= 68 - \{46 - 5(8)\}$
=gcd 6 2	$= \{100 - 2(46)\} - 46 + 5\{100 - 2(46)$
=gcd (2) 0	$= 100 - 2(46) - 46 + 5(100) - 10(46)$
	$= 6(100) - 13(46)$

$2 = 6(100) - 13(46)$

other sol^n. $2 = (100 - 13)(46) - (46 - 6)(100)$

$= 87(46) - 40(100)$

number theory:

$ax - by = \gcd(a, b)$

ex $a = 17$

$b = 7$

how gcd (a,b) = 1 $n \geq (a-1)(b-1)$

$n = ax + by$.

general.

$ax - by = gcd$

$a(x \pm nb) - b(y \mp na) = gcd.$

Special case: $(n=1)$

$ax - by = 1$

$a(x-b) - b(y-a) = gcd$

$b(a-y) - a(x-b) = gcd.$

$ax - by = 1$

$a(x-b) - b(y-a) = 1$

$\Rightarrow b(a-y) - a(b-x) = 1$

application

bully eye 7,4

role.

$gcd(7,4) = 1$

$(7-1).(4-1) = 6 \cdot 3 = 18.$

if given (a,b) with $gcd(a,b) = 1$

then for every $n \geq (a-1)(b-1)$

there exist x & $y \geq 0$ such that

$n = ax + by.$

NUMBER theory.

pierre fermat : (1600's) little theorm.

thm

for every prime number p and every integer a. (a is not divisible by p), then.

$a^{p-1} \equiv 1 \pmod p$

$a^{p-1} \equiv 1 \pmod p$

$a^4 \equiv (1 \bmod 5)$

ex $p=5$ $a=6$

$6^{5-1} = 6^4 = 216 \times 6 = 1296.$

$= 1 \pmod 5.$

- LAST theorm

$x^n + y^n = z^n$

$n > 2$

proved in 1999 wiles.

Page No.
Date

Proof $a^{p-1} - 1$ is a multiple of p.

$a \quad 2a \quad 3a \quad 4a \quad \ldots \quad (p-1)a$.

$a \cdot 2a \cdot 3a \cdots (p-1)a = 1 \cdot 2 \cdot 3 \cdots (p-1) \pmod p$

$a^{p-1}(1 \cdot 2 \cdots p-1) = (1 \cdot 2 \cdots (p-1)) \bmod p$.

$\Rightarrow a^{p-1}(p-1)! = (p-1)! \bmod p$

$a^{p-1} = 1 \bmod p$.

CRVPTOGRAPHY

a b c d e ... z

d e f g h ...

harder strategy codeword. hello → key. germans enigma.

dumbbell

public key cryptography.

PUBLIC

exponent prime: 17

choose another no. (e) → gcd $(e, p-1) = 1$

exponent prime (p): 17

enconding e = gcd e p-1 = 1 (5).

encoding

secret message = x (6) = $x^6 \bmod p$.

decoding

private (d) p = $(x^e)^d \bmod p$.

(13)

$x = x^{e \cdot d} \bmod p$. want to show

$d \cdot e = 1 \bmod p-1$ $(x^e)^d = x \bmod 17$

$5d = 1 \bmod 16$ $x^{5 \cdot 13} = x^{65} = x \bmod 17$

$a^{p-1} = 1 \bmod p$ $x^{4 \cdot 16+1} = x^{4 \cdot 16} \cdot x$

evenly divisible by 17.

$a^{p-1} = 1 \bmod p$

$x^{p-1} = 1 \bmod$ $x^{e \cdot d} = 1 \bmod p$.

$x^{4(16)} \stackrel{?}{=} 1 \pmod{17}$

$a^{p-1} = 1 \bmod p$.

fermats little theorm.

$\gg a^p = a \bmod p$.

$a^{p-1} = 1 \pmod p$

Page No.
Date

$5d = 1 \bmod 16$

CONDITIONAL STATEMENT

$p \to q$

- inverse $\sim p \to \sim q$
- converse $\sim q \to \sim p$
- contrapositive $\sim q \to \sim p$.