# Graphs!

graphs have vertices, nodes representing entities

graphs have edges to represent relationships between 2 vertices

types

directed - links go one way, not reciprocated  source → sink

undirected - all edges go both ways

weighted - each edge has weight signaling strength of relationship

unweighted - relationship strengths are equal

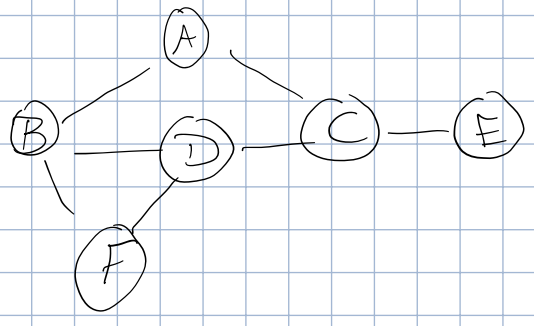
researchers co-authored at least 1 paper

undirected

can be weighted by # of papers

adjacency matrix

let's have an $N \times N$ matrix

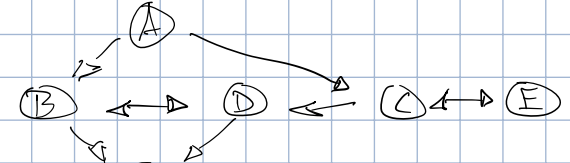



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | 0 | 1 | 1 | 0 | 0 | 0 |
| B | 1 | 0 | 0 | 1 | 0 | 1 |
| C | 1 | 0 | 0 | 1 | 1 | 0 |
| D | 0 | 1 | 1 | 0 | 0 | 1 |
| E | 0 | 0 | 1 | 0 | 0 | 0 |
| F | 0 | 1 | 0 | 1 | 0 | 0 |

table is symmetric

if it weighted, put value w

if directed, it is not symmetric

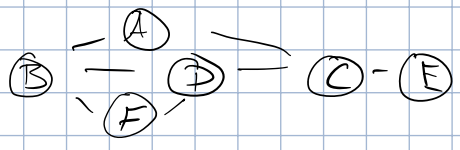



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | | | | | | |
| B | | | | | | |
| C | | | | 1 | | |
| D | 0 | 1 | 0 | 0 | 0 | 1 |

F

E
F

Adjacency list




list
A: B, C
B: A, D, F
C: A, D, E
D: B, C, F
E: C
F: B, D

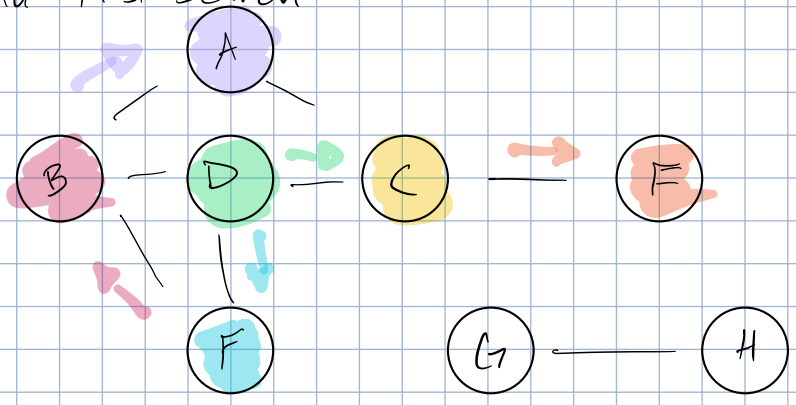
for weighted, have tuples?

A: (B, 2), (C, 1)

Part 2?

- adjacency matrix
- adjacency list

Breadth First Search




D learns secret first

Queue: D
BFS order:

who doesn't know secret?

Queue: C F B
BFS order: D

Queue: F B A E
BFS order: D C

Queue: B A E
BFS order: D C F

① look thru queue or
② B gets on queue despite already in queue. we choose this one

Queue: A E B
BFS order: D C F B

same as with B, do with A

Queue: E B A
BFS order: D C F B A

↳ nothing added to queue b/c A's neighbors have all been visited
look e n

Queue: B A
BFS order: D C F B A E

has it been visited?
has it been visited when adding to BFS?

start w/ A

Queue: A
BFS order:

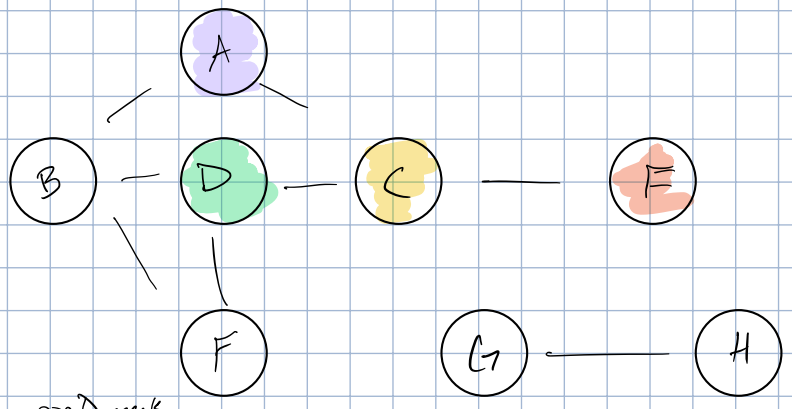
Queue: B C ← doesn't matter order
BFS order: A

3 data structures
- BFS ordered list
- visited sets — boolean
- Queue

when queue isn't empty
1. dequeue next vertex
2. skip if already visited
3. otherwise add to BFS & visited set
4. add unvisited neighbors to queue

## Depth First Search

go down rabbit hole first




stack! can't look e values

Stack: D
DFS order:
bottom → top

↳ pop D, mark as visited
push neighbors onto stack

Stack: F B C
DFS order: D

↳ pop C, mark as visited
push neighbors

Stack: F B A E
DFS order: D C

Stack : F B A
DFS order : D C E

pop E, mark.
push neighbors → but it's already visited. move on

Stack : F B B
DFS order : D C E A

pop A, mark
push neighbors — B hasn't been visited. C has

Stack : F B
DFS order : D C E A B

pop B, mark
push neighbors — F hasn't been visited

Stack : F B F
DFS order : D C E A B

popF, mark
nothing to push

Stack : F B
DFS order : D C E A B F

popB

Stack : F
DFS order : D C E A B F

popF

Stack :
DFS order : D C E A B F

order gives different DFS

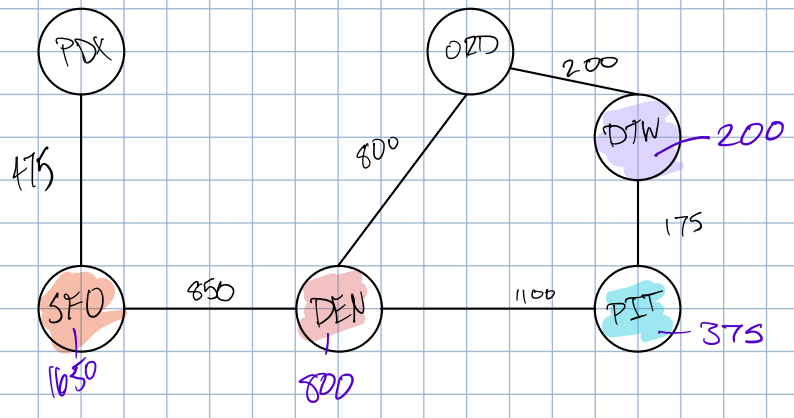
3 data structures
- DFS
- visited list
- stack

while stack isn't empty
- pop next vertex
- skip if already visited
- otherwise, add to DFS & visited set
- add (push) unvisited neighbors to stack

Part 4?

Shortest path from ORD to SFO?

strategy
- hire large # of people
- all try different routes



keep best known paths

is it done?

need priority queue PQ

PQ : ~~ORD 0~~

done w/ORD shortest from ORD to ORD is 0

PQ : DTW 0+200 , DEN 0+800

DTW 200 , DEN 800

add kids w/updated miles
not ORD b/c already know its shortest ~~dist~~
already found

PQ : DEN 800 , PIT 375

not change distance. 1475 is worse than 800

PQ : DEN 800

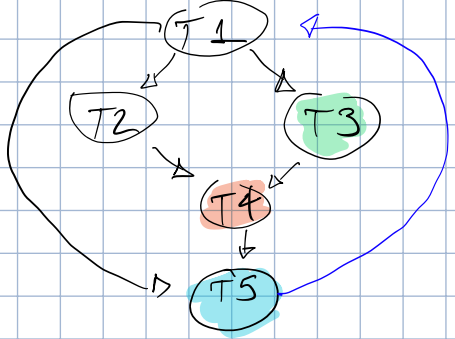
PQ : SFO 1650

starting vertex matters
get 'additional' distances for free

initialize tentative distances to ∞ & starting vertex @ 0
while heap isn't empty
remove min-distance vertex
mark as visited ; tentative distance becomes final distance
improve distances for all neighbors
add to heap or improve distance in heap already-visited neighbors can never be improved

directed acyclic graph

dependent tasks




can find ordering

if ~~★~~, it is cyclic. no ordering

DFS to check if it is cyclic

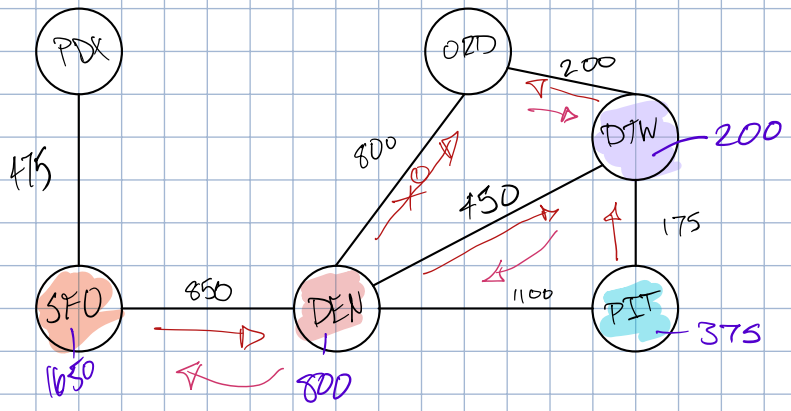
T2 → T3 ↓ | T2 → T4 ↓ | T2 → T5 ↓ | T2 → T3 ↓ ö already visited!

but when T2 → T4, it's already visited

① already ancestor
② already visited

Part 00-1

Shortest path algorithm (Dijkstra)




PQ : ~~ORD 0~~

backwards pointer

PQ : DTW 0+200 , DEN 0+800

DTW 200 , DEN 800

pointer to ORD pointer to ORD

PQ : DEN 800 , PIT 375

DEN 650 , PIT 375

650 is better than 800

change pointer to D

PQ : DEN 650 , SFO 1500

DEN 650 , SFO 1500

PQ : SFO 1500

flip pointers, gives path

# Numbers! :)

## Binary

there are unsigned numeric types

usually use for when 1 bit is needed or bit manipulation

wanna look at bit pattern? keep bytes in mind

common representation

| | base |
|---|---|
| decimal | 10 |
| binary | 2 |
| octal | 8 |
| hex | 16 |

$A3F_{16} = 10 \cdot 16^2 + 3 \cdot 16^1 + 15 \cdot 16^0 = 2623_{10}$

```
int *x = ....
printf("x: %p \n")
```
→ pointer in hex

decimal to base Y

$x = (x/Y) \cdot Y + x\%Y$

$165 \% 8 = 5$

$165 / 8 = 20$

binary & hex

$1101\ 1010\ 0111_2 = 110\{ \ 1101\{ \ 0011$

$= 6D3_{16}$

$2CE1_{16} = 0010\{ \ 1100\{ \ 1110\{ \ 0001$

$= 0010\ 1100\ 1110\ 0001_2$

15 decimal

0 32 octal

0x 8AF hex

include 0 & 0x b/c helps understand when printing

C printing nums: %u %o %x

unsigned int 032 0x8AF

bitwise operations

and: 1100 & 1010 = 1000

complement, flip bits: ~11001010 = 00110101

or: 1100 | 1010 = 1110

xor, 1, not both: 1100 ^ 1010 = 0110

shift — signed vs unsigned matters

11110011 << 1 = 11100110

11110011 >> 1 = 01111001

(01110011) >> 1

01111001

multiply & divide by 2

↳ not for 3/c division truncates

| | | | |
|---|---|---|---|
| signed | 11110011 >>1 | 11111001 | F9 |
| unsigned | 11110011 >>1 | 01111001 | 79 |

## Bitpacking

for unsigned

11110011 << 1 = 11100110

01110011 >>1 = 00111001

2-D RBG pixels in PA2

1080p 2 hour vid → 350 billion pixels

store in 0 → 255

memory needs to be divided by 4

Struct color

unsigned char red

unsigned char blue

unsigned char green

HDR stores in 6 bits per color unused → 0 → 1023 → $2^{10}-1$

$1023_{10} = 0000001111111111_2$

store in 10 bits, pack

00 RR RRRR RRRR 0000 0000 0000 0000 0000

or 1 0000 0000 0000 GGGG GGGG GG00 0000 0000

or 1 0000 0000 0000 0000 0000 00BB BBBB BBBB

00 RR RRRR RRRR GGGG GGGG GGBB BBBB BBBB

green << 10 leftshift by 10

red << 20 left shift 20

(red<<20) | or
(green<<10) |
blue

extract

00 RR RRRR RRRR GGGG GGGG GGBB BBBB BBBB

red

>> 20

0000 0000 0000 0000 0000 00 RR RRRR RRRR

blue "masking" extract specific values

& 0000 0000 0000 0000 0000 0011 1111 1111

0000 0000 0000 0000 0000 00BB BBBB BBBB

pack & 0x3FF

green

packed >>10

0000 0000 0000 RRRR RRRR RRGG GGGG GGGG

& 0000 0000 0000 0000 0000 0011 1111 1111

0000 0000 0000 0000 0000 00GG GGGG GGGG

green = (packed >> 10) & 0x3FF

signed magnitude: first bit represent sign

$$-a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i$$

| | | complement | | 2's complement |
|---|---|---|---|---|
| -4 | 1100 | -4 | 1011 | $-1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0$ |
| -3 | 1011 | -3 | | $-8 + 0 + 2 + 1 \to -5$ |
| -2 | 1010 | -2 | 1100 | |
| -1 | 1001 | -1 | 1101 | -5 1011 |
| -0 | 1001 | -0 | 1110 | |
| 0 | 0000 | 0 | 0000 | 1000 → -8 |
| 1 | 0001 | 1 | 0001 | 1111 → -1 |
| 2 | 0010 | 2 | 0010 | 0001 → 1 |
| 3 | 0011 | 3 | 0011 | |
| 4 | 0100 | 4 | 0100 | |

2 zeros

right shift signed num

1101 0010

>>1

1110 1001

-8 >> 1 → -2

-8/2 → -1

be careful mixing signed & unsigned

-1 < 1

1111 < 0001 False

need bits + context