Logistics

- Quiz and exercise 7 released tomorrow
- Quiz due Tuesday at 11:59pm
- Exercise due Friday 11:59pm
  - Just a few reductions

CS3102 Theory of Computation

www.cs.virginia.edu/~njb2b/cstheory/s2020

Warm up:

To measure the "cost" of computing something, what would units should we use?

Units

Computing Cost

- What do we actually care about with computing cost?

Notions of function “difficulty”

- Can an algorithm for function f be implemented using this computing model?
  - NAND-CIRC: answer is YES iff f is finite
  - FSA: answer is YES if function doesn’t require “memory”
  - TM: Answer is NO for $HALT_{TM}, FINITE,$ … (and many other things)
- How efficient is an algorithm for function f implemented using this computing model?
  - NAND-CIRC: How many gates?
  - FSA: (we never talked about this)
  - TM: How many transitions are required (time)? How many tape cells are required (space)?

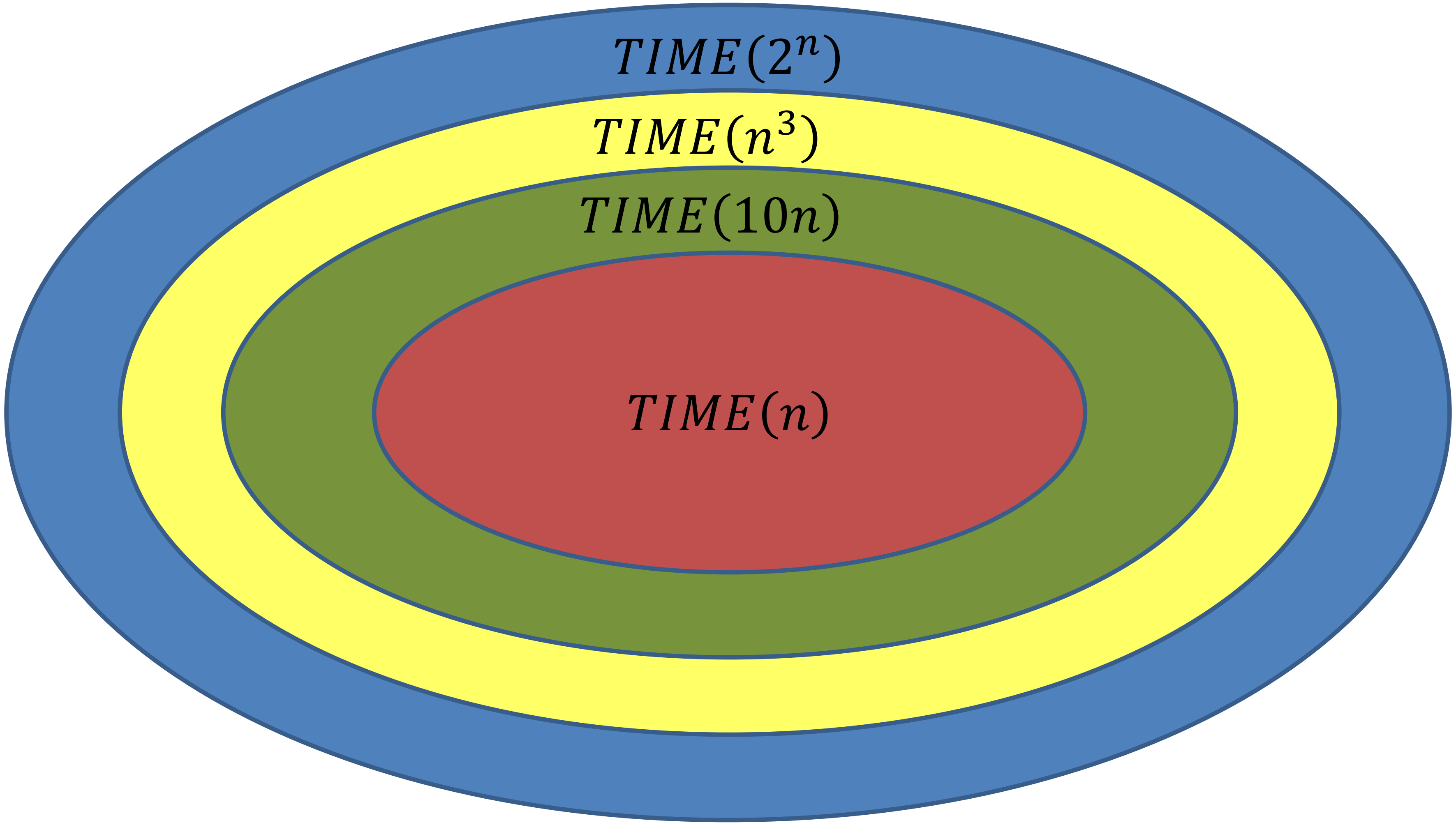
Larger inputs = More time

- Run time is not measured by a number, but a function.
- **Running time:** $T(n)$ is a function mapping naturals to naturals. We say $F: \{0,1\}^* \rightarrow \{0,1\}^*$ is computable in $T(n)$ time if there exists a TM M s.t. for every large n and ever input $x \in \{0,1\}^n$, M halts after at most $T(n)$ steps and outputs $F(x)$.
- $TIME\big(T(n)\big)$ represents the set of boolean functions computable within $T(n)$ time

Examples

- How long will XOR take on a Turing Machine?
  - We have an even state and an odd state
  - For each bit, move right, switch states if 1
  - Halt when you get to end of input
- How long will MAJ take on a Turing Machine?
  - Find a zero, cross it off
  - Go to beginning
  - Find a one, cross it off
  - Go to beginning
  - Halt when no more 0s or no more 1s

More time gives more functions

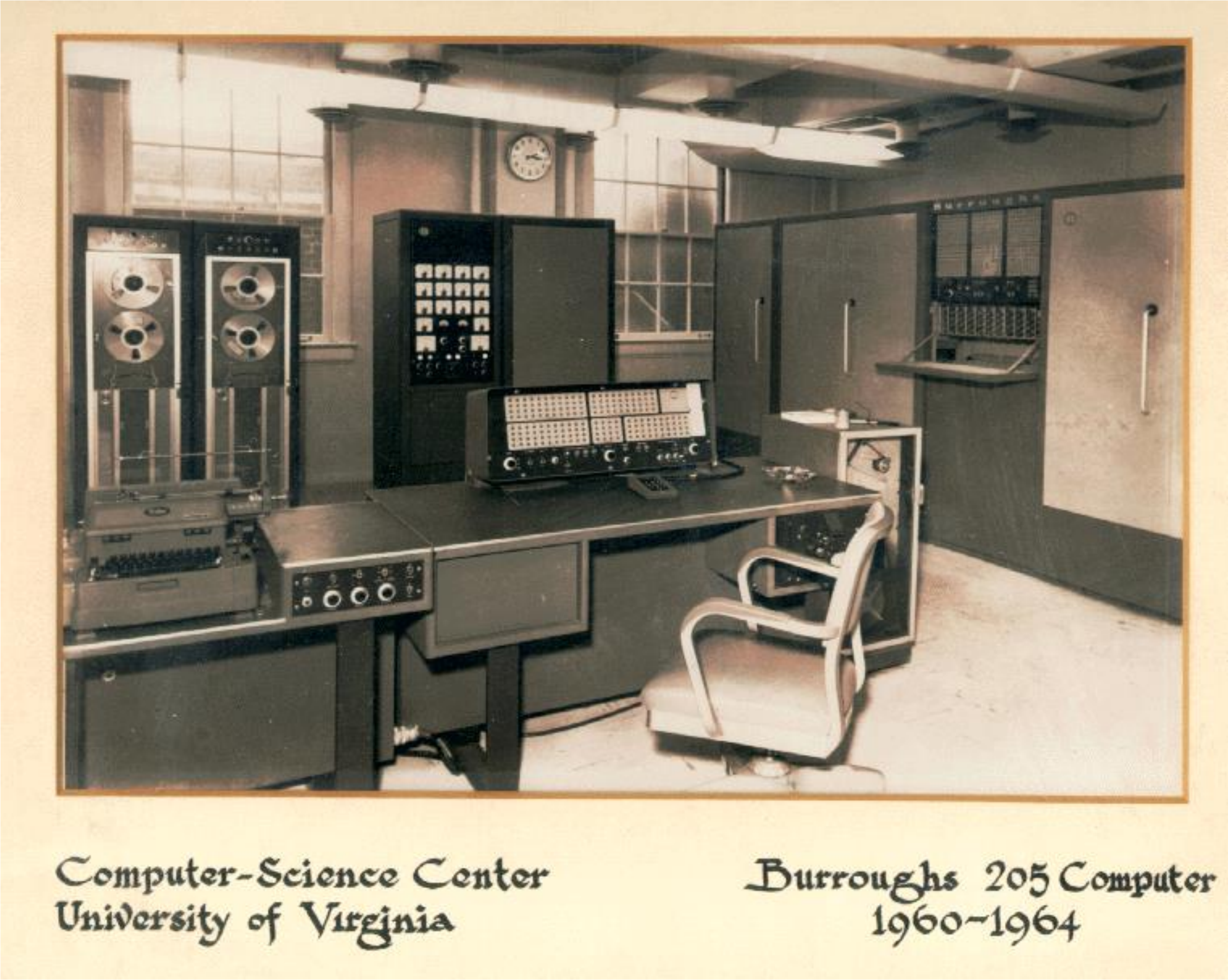


Different computing Models may have Different Running Times

- So far: a TM uses a tape. Can only visit a neighboring cell from the current one.

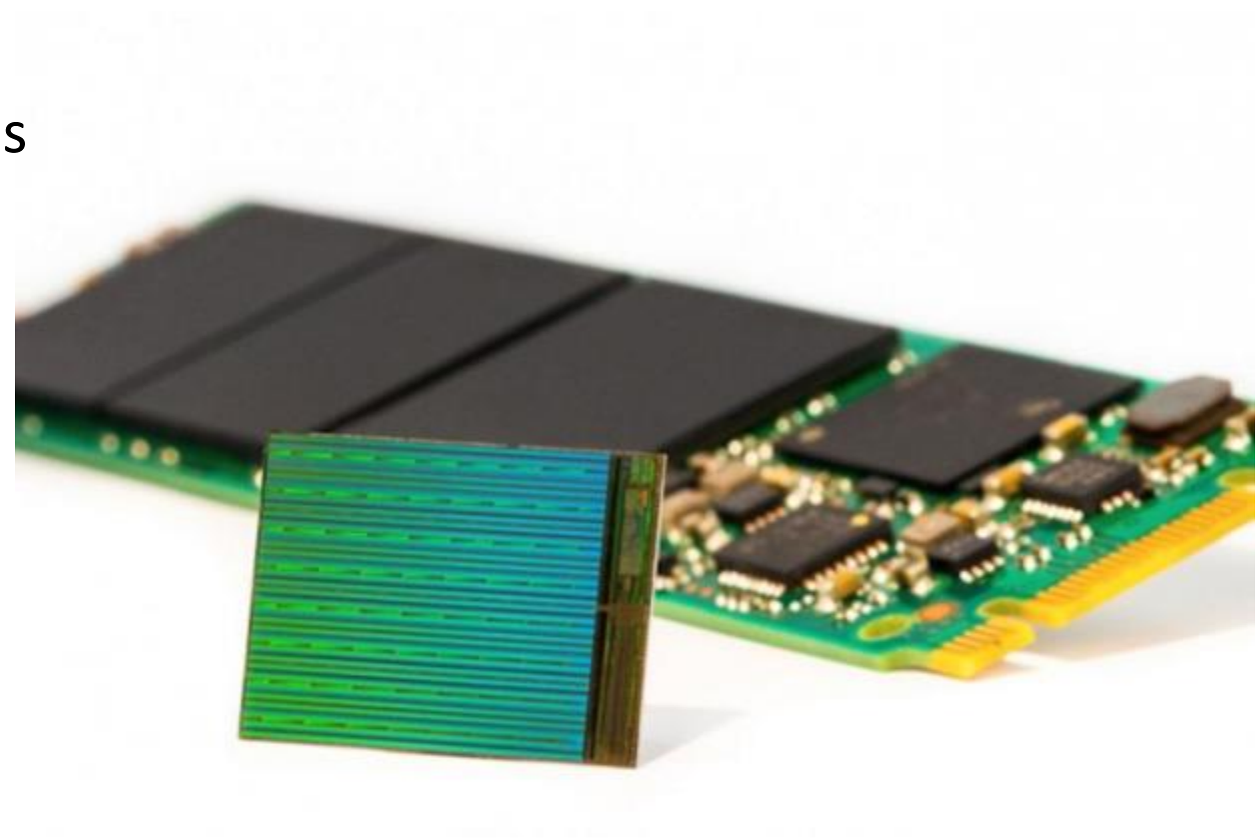
1960s

A tape was probably a reasonable memory model



Today

Can look up two locations without visiting all locations between.
"Random" access (RAM)



RAM Machine

- We can go directly to a certain index in the tape

To transition:

1. Have a second tape to keep track of current location (increment each time we move right, decrement for left)
2. Have a third tape to record the target location
3. Move until the two tapes match
    1. Maybe we need another tape to do this?

(details not important, but if you want them, see 7.2 in text)

Important observation: Tape-machine takes more steps than a RAM machine (if a RAM-TM computes f in $T(n)$ time, a tape TM can compute f in $\left(T(n)\right)^4$ time, see theorem 12.5 for details)

Finding Running Times

- We will find running times for the following:
  - Shortest Path in a graph
  - Longest Path in a graph
  - 3SAT
  - 2SAT

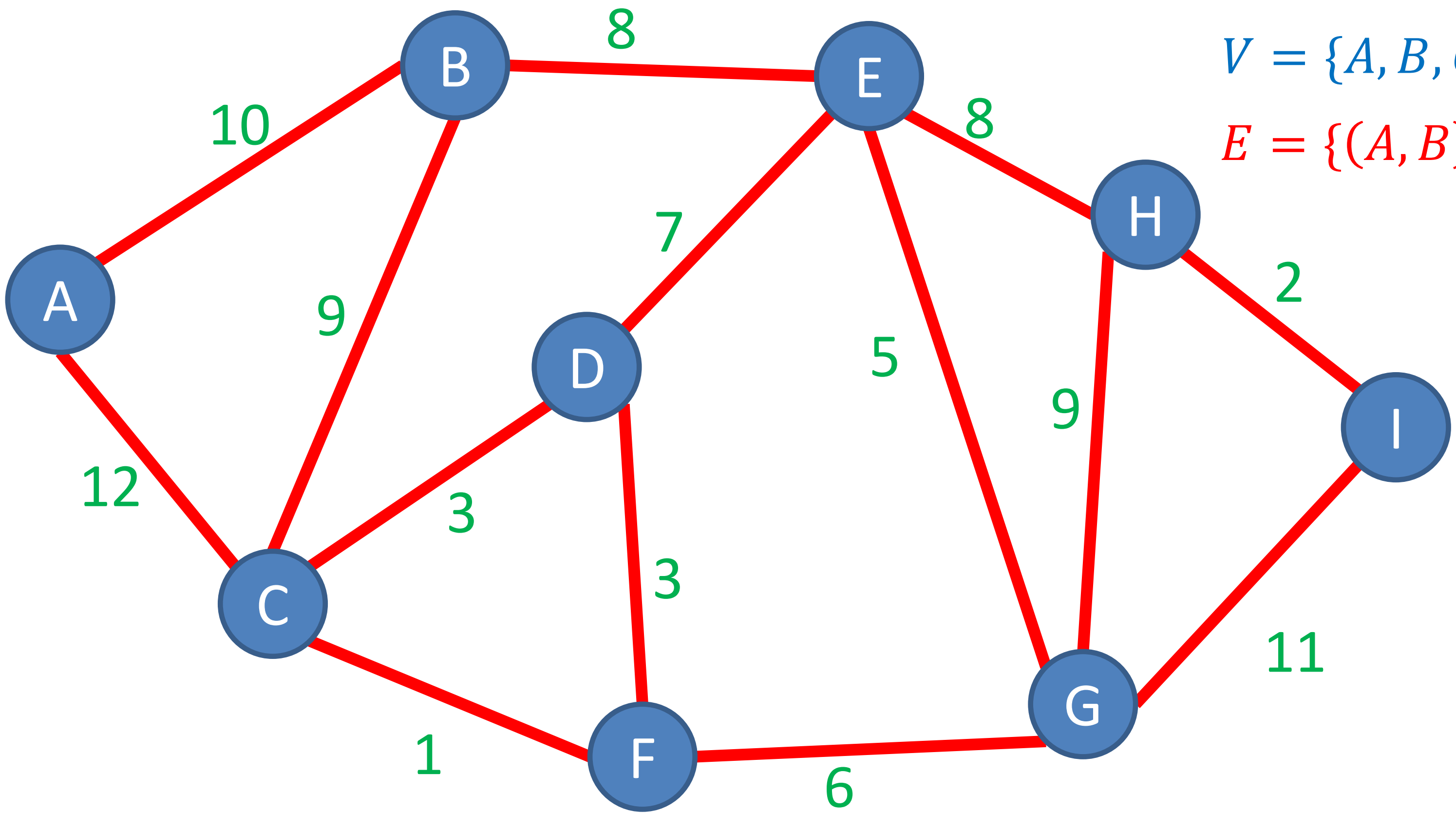
Graphs

Vertices/Nodes

Definition: $G = (V, E)$

Edges

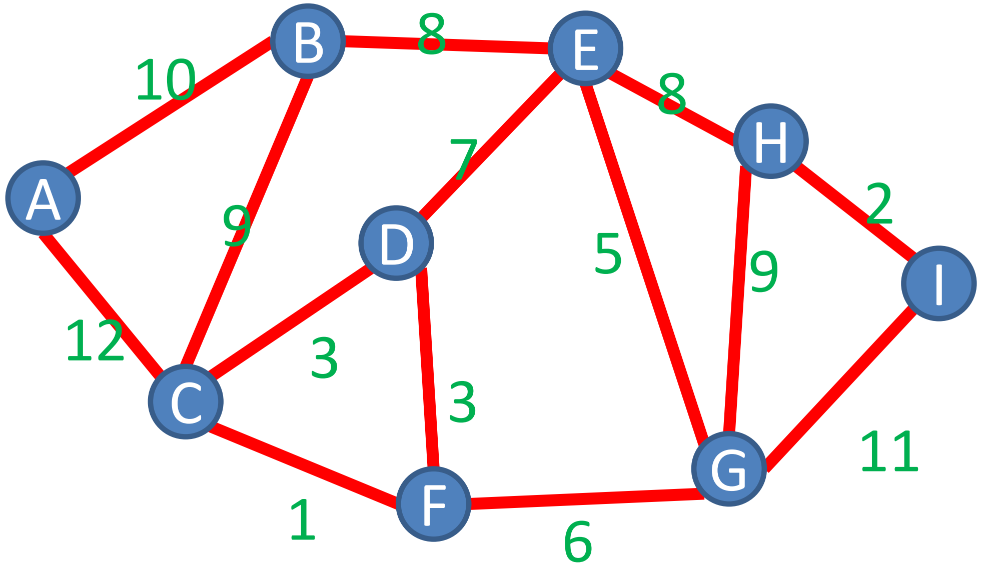
$w(e)$ = weight of edge e



$V = \{A, B, C, D, E, F, G, H, I\}$

$E = \{(A, B), (A, C), (B, C), \ldots\}$

Adjacency List Representation



A	B	C		
B	A	C	E	
C	A	B	D	F
D	C	E	F	
E	B	D	G	H
F	C	D	G	
G	E	F	H	I
H	E	G	I	
I	G	H		

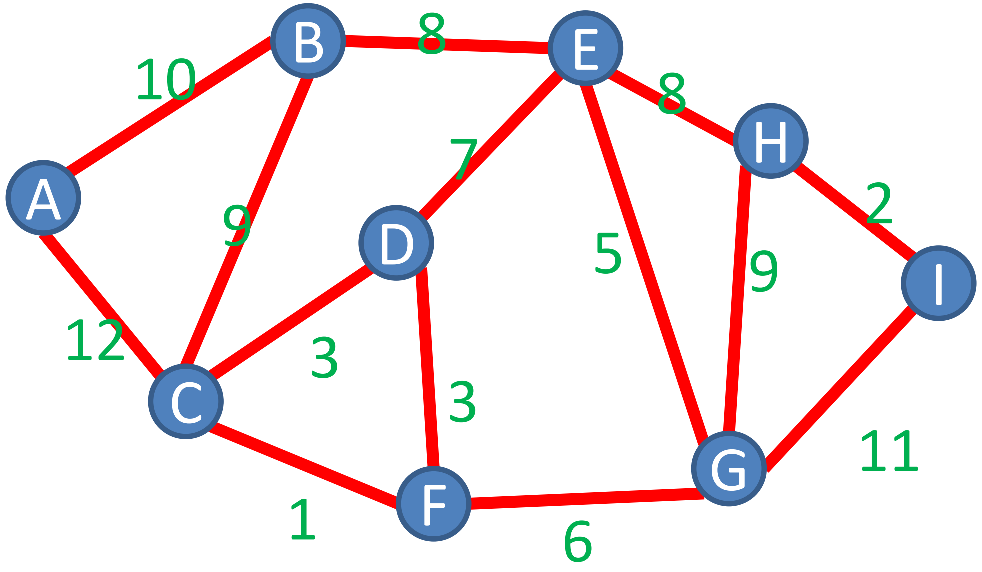
Tradeoffs

Space: $V + E$

Time to list neighbors: $Degree(A)$

Time to check edge (A, B): $Degree(A)$

Adjacency Matrix Representation



	A	B	C	D	E	F	G	H	I
A		1	1						
B	1		1		1				
C	1	1		1		1			
D			1		1	1			
E		1		1			1	1	
F			1	1			1		
G					1	1		1	1
H					1		1		1
I							1	1	

<u>Tradeoffs</u>

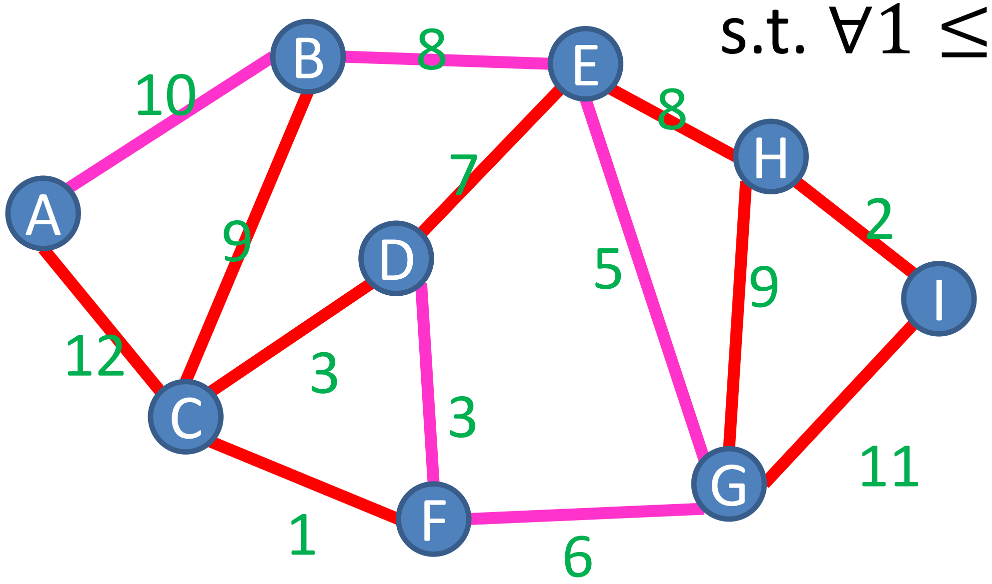
Space: V^2

Time to list neighbors: V

Time to check edge (A,B):$O(1)$

Definition: Path

A sequence of nodes $(v_1, v_2, \dots, v_k)$
s.t. $\forall 1 \leq i \leq k-1$, $(v_i, v_{i+1}) \in E$

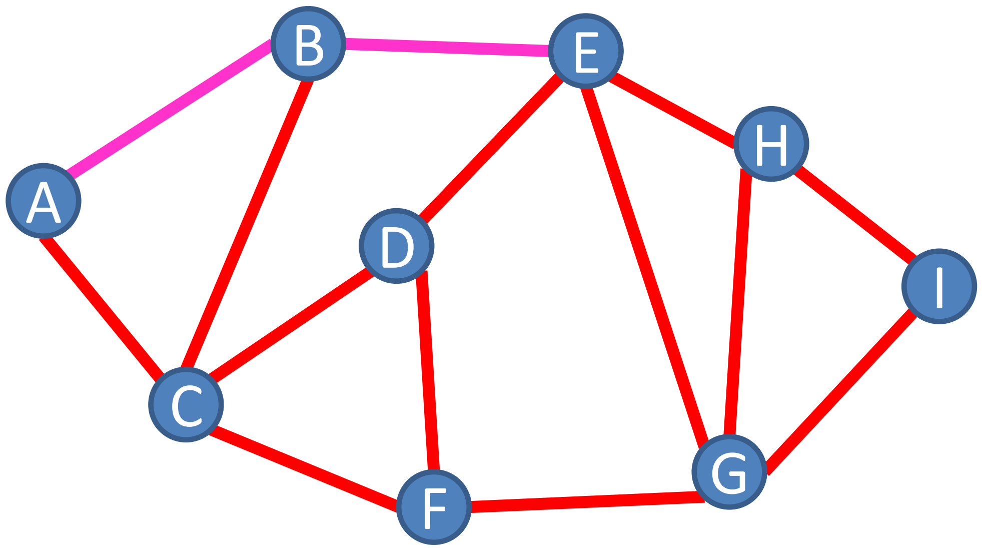


Simple Path:
A path in which each node appears at most once

Cycle:
A path of > 2 nodes in which $v_1 = v_k$

Shortest Path

- Given an unweighted graph, start node s and an end node t, how long is shortest path from s to t?



Shortest path from A to E has length 2

Breadth First Search

Find a path from s to t

Keep a queue Q of nodes

$hops = 0$

$Q.enqueue((s, hops))$

While Q is not empty and $v \, != t$:

$\quad v, hops = Q.dequeue()$

$\quad$ for each "unvisited" $u \in V$ s.t. $(v, u) \in E$:

$\quad\quad Q.enqueue((u, hops + 1))$

Running time: $O(|V| + |E|)$

Longest Path

- Given a start node s and an end node t, how long is longest path from s to t?

Longest Path

Enumerate all possible sequences of nodes
check if it's a path
print the length of the longest one

Running time: $n!$