Algorithms

Graphs:
intro + terminology

Recap

- Oral grading today + tomorrow
- Review on Wed.
  ↳ bring questions!
- Test Friday
- Readings due next week
- HW5 due after break

Greedy algs:

The key is finding how to be greedy.

Proving correctness:

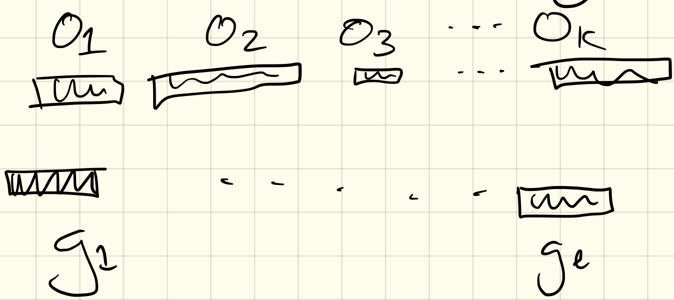
1. Find some key property, & show greedy was it.

   Ex: 4.1 on File Storage:

   cost was minimized if $L[i] \le L[i+1]$

2. Compare opt to greedy & show you can swap.

   Ex: 4.2 on Scheduling



Post midterm:

On to graphs!

Today will be a basic overview

Reading (Sec 5.2 + 5.3) should go quickly – due next Monday

I mostly want to set up basics of notation + key results...

(Note: Will skip ch. 6 entirely.)

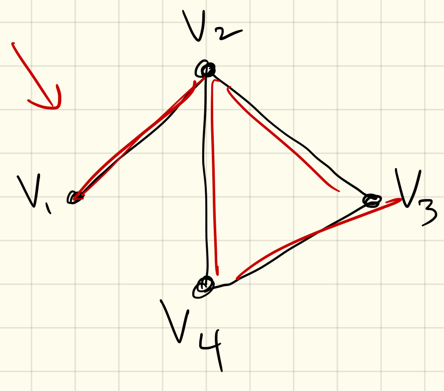
Graphs

A graph $G = (V, E)$ is an ordered pair of 2 sets:

V = vertices $= \{v_1, v_2, v_3, v_4\}$

E = edges $= \{\{v_1, v_2\}, \{v_2, v_3\}, \dots\}$

View:



Why?

They model everything!

Examples

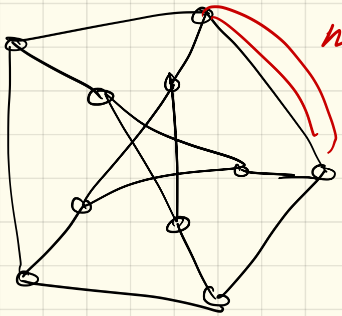
⋮

LOTS

More dfns:

G is simple undirected if edges are unordered pairs

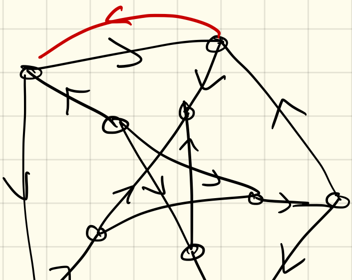
so $\{u,v\} = \{v,u\}$

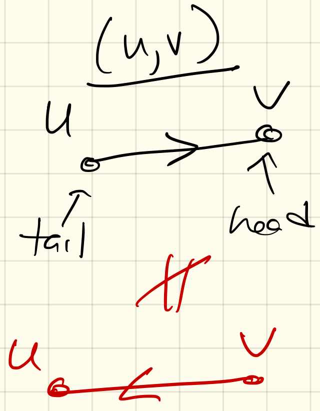


multiedges: non-simple

G is directed if edges are ordered pairs

so $(u,v) \neq (v,u)$





Dfns cont:

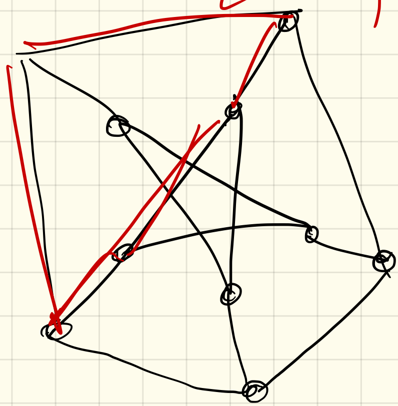
The <u>degree</u> of a vertex, $d(v)$, is ~~the~~ number of ~~adjacent~~ incident edges.

A <u>path</u> $P = v_1, \ldots, v_k$ is a set of vertices with $\{v_i, v_{i+1}\} \in E$ (or $(v_i, v_{i+1}) \in E$ if directed),

A path is <u>simple</u> if all vertices are distinct

A cycle is a path which is simple except $v_1 = v_k$.

Simple cycle



walk: repeated vertices & edges

circuit: cycle but allows repeats

Lemma: (degree-sum formula)

$$\sum_{v \in V} d(v) = 2|E|$$

pf: - Every edge has 2 endpoints.

Size of G:

2 parameters:

$|V| = n$

$|E| = m$.

How big can m be in terms of n?

(simple graph)

D-S form: $d(v_1)+d(v_2)+\cdots+d(v_n)=m$

$$m = O(n^2)$$

Each vertex can be connected to n-1 others.

$$m \le \underset{v_1}{(n-1)} + \underset{v_2}{(n-2)} + \underset{v_3}{(n-3)} + \cdots + 1$$

$$= \sum_{i=1}^{n-1} i = \binom{n}{2} = \frac{n(n-1)}{2}$$

Tree: $m = n-1$

Representing graphs

How do we make this data structure?

- lists
- matrix

$\Bigg\}$ tradeoffs

Adjacency (or vertex) lists:

Each vertex:

n lists

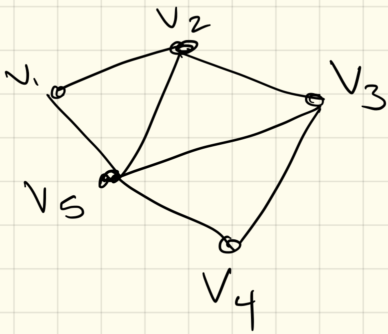
v_1 : v_2, v_5

v_2 : v_1, v_3, v_5

v_3 : v_2, v_4, v_5

v_4 : v_3, v_5

v_5 : $\underbrace{v_1, v_2, v_3, v_4}_{2|E|}$



Size: $O(n+m)$

Lookup: Time to check if v_i + v_j are nbrs:

check in a list

linked: $O(n)$ (rather $d(v_i)$)

array: $O(\log n)$

Implementation:

More buried data structures!

Could use:

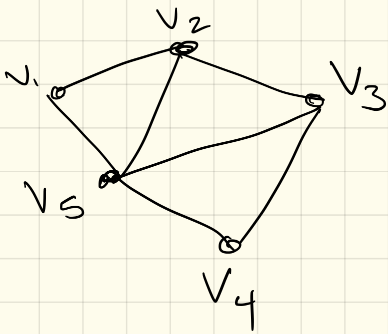
→ linked

or

array-based

Adjacency Matrix

	v_1	v_2	v_3	v_4	v_5
v_1		1	0	0	1
v_2			1	0	1
v_3				1	1
v_4					1
v_5					



use if G is directed

space : $O(n^2)$

check nbr: $O(1)$

~~Incidence matrix:~~

$$\cancel{\begin{array}{c|ccc} & e_1 & \cdots & e_m \\ \hline v_1 & & & \\ \vdots & & & \\ v_n & & & \end{array}}$$

Which is better?

Depends!

	Adjacency matrix	Standard adjacency list (linked lists)	Adjacency list (hash tables)
Space	$\Theta(V^2)$	$\Theta(V+E)$	$\Theta(V+E)$
Time to test if $uv \in E$	$O(1)$	$O(1+\min\{\deg(u),\deg(v)\}) = O(V)$	$O(1)$
Time to test if $u \to v \in E$	$O(1)$	$O(1+\deg(u)) = O(V)$	$O(1)$
Time to list the neighbors of v	$O(V)$	$O(1+\deg(v))$	$O(1+\deg(v))$
Time to list all edges	$\Theta(V^2)$	$\Theta(V+E)$	$\Theta(V+E)$
Time to add edge uv	$O(1)$	$O(1)$	$O(1)^*$
Time to delete edge uv	$O(1)$	$O(\deg(u)+\deg(v)) = O(V)$	$O(1)^*$

In the rest of this book, unless explicitly stated otherwise, all time bounds for graph algorithms assume that the input graph is represented by a standard adjacency list. Similarly, unless explicitly stated otherwise, when an exercise asks you to design and analyze a graph algorithm, you should assume that the input graph is represented in a standard adjacency list.

Next time

(in one week):

BFS + DFS quick recap

Then onto MST + shortest paths.