DSC 40B

Theoretical Foundations II

Lecture 10 | Part 1

News

DSC 40B

Theoretical Foundations II

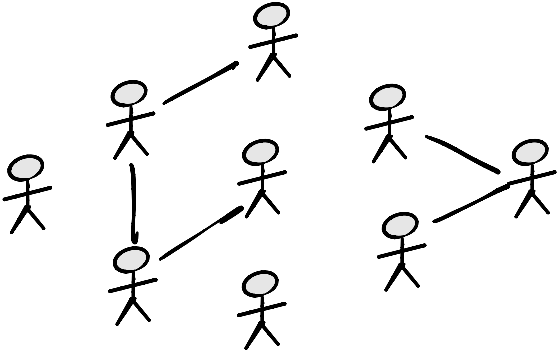
Lecture 10 | Part 2

Graphs

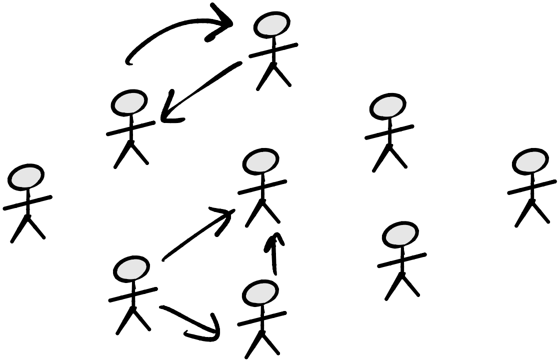
Data Types

- **Feature vectors**
  - We care about attributes of individuals.
- **Graphs**
  - We care about relationships between individuals.

Example: Facebook



Example: Twitter



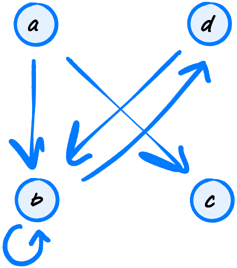
Definition

A **directed graph** (or **digraph**) G is a pair (V, E) where V is a finite set of **nodes** (or **vertices**) and E is a set of ordered pairs (the **edges**).

Example:

$V = \{a, b, c, d\}$

$E = \{(a, c), (a, b), (d, b), (b, d), (b, b)\}$



Directed Graphs (More Formally)

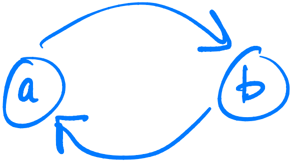
E is a subset of the **Cartesian product**, $V \times V$.

Example:
$\{a, b, c\} \times \{1, 2\} =$ $\{(a,1), (a,2), (b,1), (b,2), (c,1), (c,2)\}$

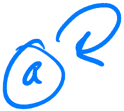
Consequences

Because the edge set of a directed graph is allowed to be *any* subset of $V \times V$:

- the edges have directions.
  - e.g., (a, b) is "from a to b"



- can have "opposite" edges.
  - e.g., (a, b) and (b, a).
- can have "self-loops"
  - e.g., (a, a)

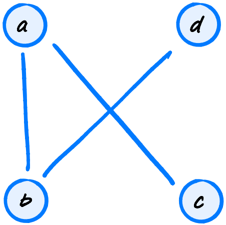


Definition

An **undirected graph** G is a pair (V, E) where V is a finite set of **nodes** (or **vertices**) and E is a set of unordered, distinct pairs (the **edges**).

Example:

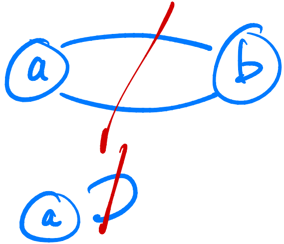
$$V = \{a, b, c, d\}$$
$$E = \{\{a, c\}, \{a, b\}, \{d, b\}\}$$



Undirected Graphs (More Formally)

An edge in an undirected graph is a set $\{u, v\}$ where $u \neq v$. This has consequences:

- the edges have **no direction**.
  - e.g., $\{a, b\}$ is **not** “from” a “to” b.
- **cannot** have “opposite” edges.
  - e.g., $\{a, b\}$ and $\{b, a\}$ are the same.
- **cannot** have “self-loops”
  - e.g., $\{a, a\}$ is not a valid edge



Notational Note

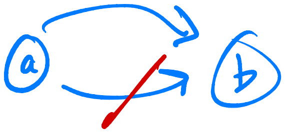
Although edges in undirected graphs are sets, we typically write them as pairs: (u, v) instead of $\{u, v\}$.

Summary

- Edges have direction:
  - Directed: **yes**
  - Undirected: **no**
- Self-loops, (u, u)?
  - Directed: **yes**
  - Undirected: **no**
- Opposite edges, (u, v) and (v, u)?
  - Directed: **yes**
  - Undirected: **no** (they are the same edge)

Note

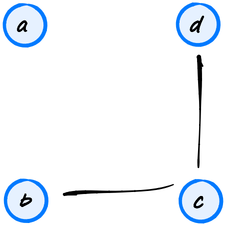
Neither directed nor undirected graphs can have **duplicate edges**[1]



[1] There are other definitions which allow duplicate edges.

Note

Graphs don't need to be "connected"[2]



[2] There are other definitions which allow duplicate edges.

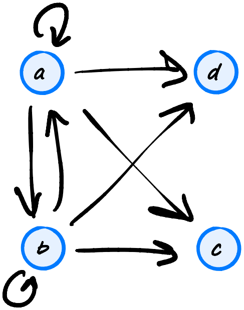
Exercise

What is the greatest number edges possible in a **directed** graph?

Counting Edges

$n = |V|$

What is the greatest number edges possible in a **directed** graph?



$4+4+4+4$

$= 4^2 = 16$

$\boxed{n^2}$ or $\boxed{|V|^2}$

Exercise

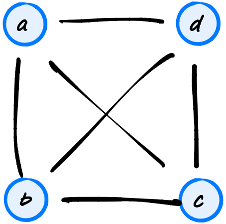
What is the greatest number edges possible in an **undirected** graph?

Counting Edges

$$\frac{|V|(|V|-1)}{2} = \binom{|V|}{2}$$

What is the greatest number edges possible in an **undirected** graph?

$3+2+1=6$



$1+2+3+\dots+(n-1)$

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

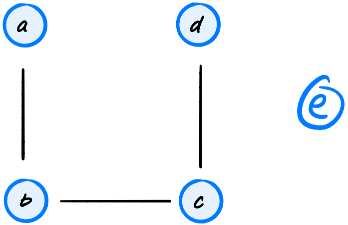
$$= \Theta(n^2) = \Theta(V^2)$$

Degree

The **degree** of a node in an undirected graph is the number of edges containing that node.

degree(b) = 2

degree(a) = 1



In-Degree/Out-Degree

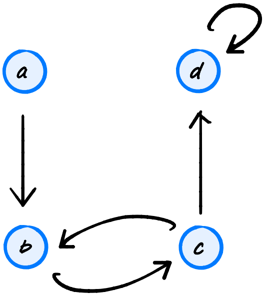
The **in-degree** of a node in an directed graph is the number of edges **entering** that node.

The **out-degree** of a node in an directed graph is the number of edges **leaving** that node.

The **degree** of a node in a directed graph is the in-degree + out-degree.

Examples

in-degree(b) = 2
out-degree(b) = 1
degree(b) = 2+1
= 3

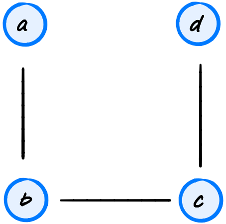


in-degree(d) = 2
out-degree(d) = 1
degree(d) = 3

Neighbors

Definition: in an undirected graph, the set of **neighbors** of a node *u* is the set of all nodes which share an edge with *u*.

neighbors(b) = $\{a, c\}$

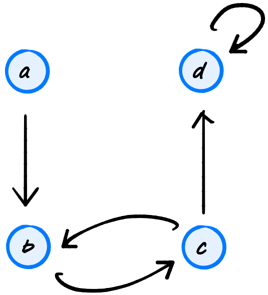


neighbors(a) = $\{b\}$

Predecessors

Definition: in an directed graph, the set of **predecessors** of a node *u* is the set of all nodes which are at the **start** of an edge **entering** *u*.

predecessors(b) = $\{a, c\}$

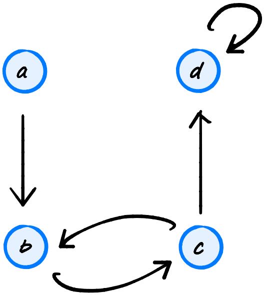


predecessors(d) = $\{c, d\}$

Successors

Definition: in an directed graph, the set of **successors** of a node *u* is the set of all nodes which are at the **end** of an edge **leaving** *u*.

successors(b) = $\{c\}$

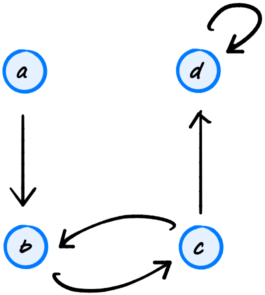


successors(d) = $\{d\}$

A Convention

In a directed graph, the **neighbors** of u are the **successors** of u.

.neighbors( )



DSC 40B

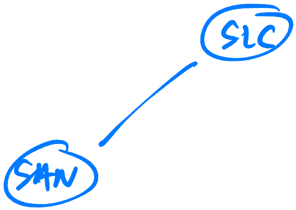
Theoretical Foundations II

Lecture 10 | Part 3

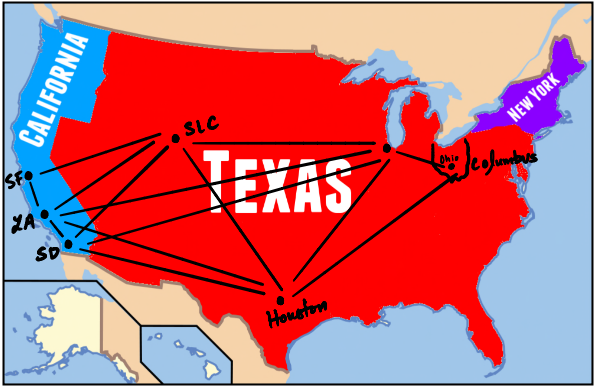
Paths

Example

- Consider a graph of direct flights.
- Each node is an airport.
- Each edge is a direct flight.
- Should the graph be directed or undirected?



Example



CALIFORNIA
NEW YORK
TEXAS
SLC
SF
LA
SD
Ohio
Columbus
Houston

Example

- Can we get from San Diego to Columbus?
- Not with a single edge.
- But with a **path**.

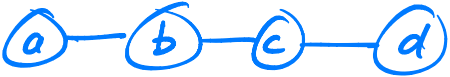
Definition

A **path** from u to u' in a (directed or undirected) graph $G = (V, E)$ is a sequence of one or more nodes $u = v_0, v_1, \dots, v_k = u'$ such that there is an edge between each consecutive pair of nodes in the sequence.

Path Length

Definition: The **length** of a path is the number of nodes in the sequence, minus one. Paths of length zero are possible!

(a, b, c, d)



(a)

Examples

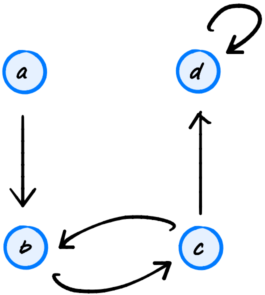
Yes

(a, b, c, d)

(a, b, c, b, c)

(d, d, d, d, d)

(c)



No

(a, c, b)

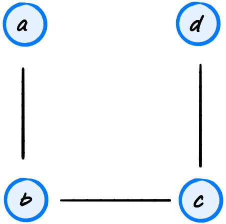
Examples

Yes

(a, b, c, d)

(a, b, c, b, c, b, c)

(c)



No

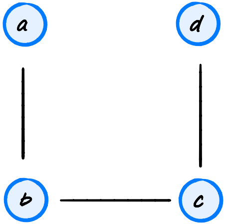
(a, c, b)

(d, d, d, d, d)

Note

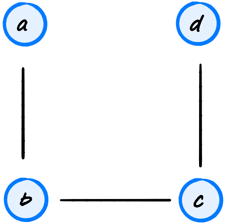
Paths **can** go through the same node more than once!

(a, b, c, b, c, b, c, d)



Simple Paths

Definition: A **simple path** is a path in which every node is unique.



Reachability

Definition: node v is **reachable** from node u if there is a path from u to v.

Reachability and Directedness

- If G is undirected, reachability is symmetric.
  - If u reachable from v, then v reachable from u.
- If G is directed, reachability is **not** symmetric.
  - If u reachable from v, then v may/may not be reachable from u.

Important Trivia

- In any graph, any node is **reachable** from **itself**.

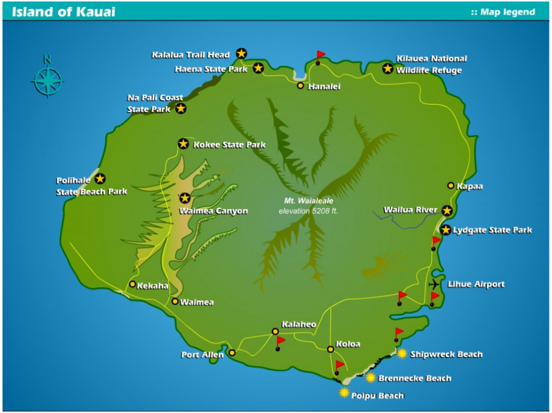
DSC 40B

Theoretical Foundations II

Lecture 10 | Part 4

Connected Components

Example



Island of Kauai
:: Map legend
N
Kalalua Trail Head
Haena State Park
Hanalei
Kilauea National
Wildlife Refuge
Na Pali Coast
State Park
Kokee State Park
Polihale
State Beach Park
Waimea Canyon
Mt. Waialeale
elevation 5208 ft.
Kapaa
Wailua River
Lydgate State Park
Lihue Airport
Kekaha
Waimea
Kalaheo
Koloa
Port Allen
Shipwreck Beach
Brennecke Beach
Poipu Beach

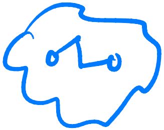
Example


Hawaii
KAUA'I
NI'IHAU
O'AHU
Honolulu
Pearl City
Kāne'ohe
Kailua
MOLOKA'I
LĀNA'I
MAUI
KAHO'OLAWE
HAWAI'I
Hilo
Kailua-Kona
Kaua'i Channel
Kaiwi Channel
'Alenuihāhā Channel
Kaulakahi Channel
OnTheWorldMap.com

Connectedness

A graph is **connected** if every node u is reachable from every other node v. Otherwise, it is **disconnected**.

Equivalent: there is a path between every pair of nodes.




Connected Components

A **connected component** is a maximally-connected set of nodes.

I.e., if $G = (V, E)$ is an undirected graph, a connected component is a set $C \subset V$ such that

- any pair $u, u' \in C$ are reachable from one another; and
- if $u \in C$ and $z \notin C$ then u and z are not reachable from one another.

Exercise

What are the connected components?

$$V = \{0, 1, 2, 3, 4, 5, 6\}$$
$$E = \{(0, 2), (1, 5), (3, 1), (2, 4), (0, 4), (5, 3)\}$$

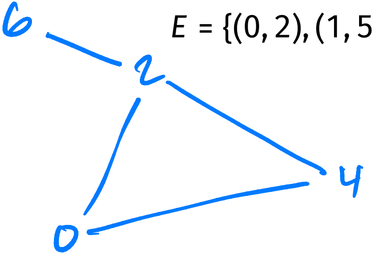
Example

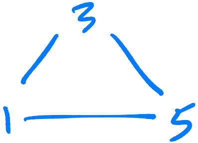
$\{0, 2, 4, 6\}$
$\{1, 3, 5\}$,

What are the connected components?

$V = \{0, 1, 2, 3, 4, 5, 6\}$

$E = \{(0, 2), (1, 5), (3, 1), (2, 4), (0, 4), (5, 3)\}$, $(2, 6)$





DSC 40B

Theoretical Foundations II

Lecture 10 | Part 5

Adjacency Matrices

Representations

- How do we **store** a graph in a computer's memory?

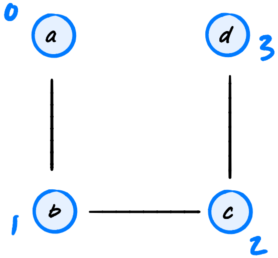
- Three approaches:
  1. Adjacency matrices.
  2. Adjacency lists.
  3. "Dictionary of sets"

Adjacency Matrices

- Assume nodes are numbered 0, 1, ..., $|V| - 1$
- Allocate a $|V| \times |V|$ (Numpy) array
- Fill array as follows:
  - `arr[i,j] = 1` if $(i,j) \in E$
  - `arr[i,j] = 0` if $(i,j) \notin E$

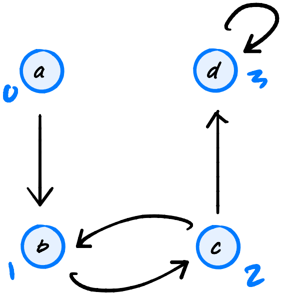
$arr[0,1]$

Example



$$
\begin{array}{c|cccc}
 & 0 & 1 & 2 & 3 \\
\hline
0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 \\
2 & 0 & 1 & 0 & 1 \\
3 & 0 & 0 & 1 & 0
\end{array}
$$

Example



$$\begin{array}{c|c} & \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Observations

- If G is undirected, matrix is symmetric.
- If G is directed, matrix may not be symmetric.

Time Complexity

operation[3]	code	time
edge query	`adj[i,j] == 1`	$\Theta(1)$
degree(i)	`np.sum(adj[i,:])`	$\Theta(\lvert V \rvert)$

[3]For undirected graphs

Space Requirements

- Uses $|V|^2$ bits, even if there are very few edges.
- But most real-world graphs are **sparse**.
  - They contain many fewer edges than possible.

Example: Facebook

- Facebook has 2 billion users.

$$
\begin{aligned}
(2 \times 10^9)^2 &= 4 \times 10^{18} \text{ bits} \\
&= 500 \text{ petabits} \\
&\approx 6500 \text{ years of video at 1080p} \\
&\approx 60 \text{ copies of the internet as it was in 2000}
\end{aligned}
$$

Adjacency Matrices and Math

- Adjacency matrices are useful mathematically.
- Example: (i, j) entry of A^2 gives number of hops of length 2 between i and j.

DSC 40B
Theoretical Foundations II

Lecture 10 | Part 6

Adjacency Lists

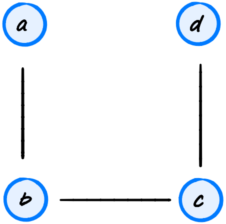
What's Wrong with Adjacency Matrices?

- Requires $\Theta(|V|^2)$ storage.
- Even if the graph has no edges.
- **Idea**: only store the edges that exist.

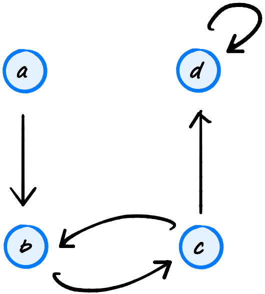
Adjacency Lists

- Create a list `adj` containing $|V|$ lists.
- `adg[i]` is list containing the neighbors of node i.

Example



Example



Observations

- If G is undirected, each edge appears twice.
- If G is directed, each edge appears once.

Time Complexity

operation[4]	code	time
edge query	`j in adj[i]`	$\Theta(\text{degree}(i))$
degree(i)	`len(adj[i])`	$\Theta(1)$

[4]For undirected graphs

Space Requirements

- Need $\Theta(|V|)$ space for outer list.
- Plus $\Theta(|E|)$ space for inner lists.
- In total: $\Theta(|V| + |E|)$ space.

Example: Facebook

- Facebook has 2 billion users, 400 billion friendships.

- If each edge requires 32 bits:

$$
\begin{aligned}
&(2 \text{ bits} \times 200 \times (2 \text{ billion}) \\
&= 64 \times 400 \times 10^9 \text{ bits} \\
&= 3.2 \text{ terabytes} \\
&= 0.04 \text{ years of HD video}
\end{aligned}
$$

DSC 40B

Theoretical Foundations II

Lecture 10 | Part 7

Dictionary of Sets

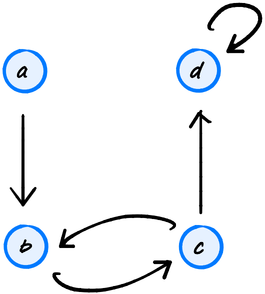
Tradeoffs

- Adjacency matrix: fast edge query, lots of space.
- Adjacency list: slower edge query, space efficient.
- Can we have the best of both?

Idea

- Use **hash tables**.
- Replace inner edge lists by `set`s.
- Replace outer list with `dict`.
  - Doesn't speed things up, but allows nodes to have arbitrary labels.

Example



Time Complexity

operation[5]	code	time
edge query	`j in adj[i]`	$\Theta(1)$ average
degree(i)	`len(adj[i])`	$\Theta(1)$ average

[5] For undirected graphs

Space Requirements

- Requires only $\Theta(E)$.
- But there is overhead to using hash tables.

Dict-of-sets implementation

- Install with `pip install dsc40graph`
- Import with `import dsc40graph`
- Docs: `https://eldridgejm.github.io/dsc40graph/`
- Source code: `https://github.com/eldridgejm/dsc40graph`
- Will be used in HW coding problems.