Learning Goals

- Technically: Distinguish when BFS vs. Dijkstra applies; How to implement Dijkstra.
- Civically: See how 'weights' encode *whose* costs or barriers matter in real life (influence, trust, access, language, bandwidth, safety).
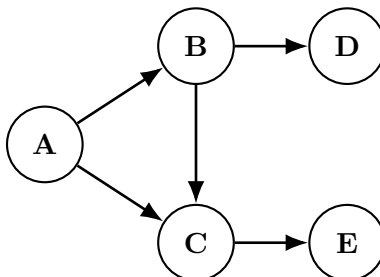
Graph Review

1. An **unweighted** graph treats every edge with the same <u>cost (often 1)</u>.

2. A **weighted** graph assigns each edge a <u>numerical weight (time, cost, risk, etc.)</u>.

3. A **directed** graph has edges with <u>directions</u> (one-way links).

4. An **undirected** graph has edges usable in <u>both</u> directions.

Shortest Path Problem in Social Network

In a social network or communication system, the shortest path represents the most efficient route for information to travel between people or groups, often in terms of cost.

A. BFS and Unweighted Graph

Here, nodes represent 5 people online; edges are chances they connect and share information. A post starts at node **A**. Level refers to the depth or distance of a node from the starting node.

Task. Using BFS from source **A**, list the first time each node 'sees' the post (fewest shares). Fill discovery order and parent pointers:

Vertex v	Ava (A)	Bob (B)	Chloe (C)	Dylan (D)	Emma (E)
Level (steps from A)	0	1	1	2	2
BFS parent	–	Ava	Ava	Bob	Chloe

Shortest (fewest shares) path from A to E: <u>Ava → Chloe → Emma</u>

> **Civic Reflection (discuss in groups)** In this *equal–step* model, who gets the post earliest and why? Who might be disadvantaged if some connections do not exist (rural users, language communities, people off major platforms)?
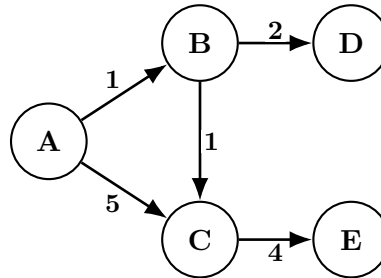
Give one concrete real-world advantage of using BFS here, and one risk of assuming all edges are equally easy to use.

- Advantage: It is simple and finds the fewest handoffs, which is good when every share or call takes about the same effort.
- Risk: It hides barriers like bad internet, language gaps, or safety concerns and pretends all connections are equally easy to use.

> **Bridge Idea.** BFS assumes *equality of access*: every edge costs '1 share' or "1 call." Real life adds frictions: language, bandwidth, attention, safety, trust, time, money.

Part B. Dijkstra and Weighted 'Influence/Trust' Graph

Now we add edge weights to encode influence / trust / effort. Weight represents difficulty for information to pass through that direction. A larger weight means it is *harder* or *costlier* for information to travel that edge (e.g., low trust, language barrier, limited data plan). Smaller weight = easier spread. With the same 5 people, imagine Ava is posting to defend against a rumor or raise awareness about issues.

Task Run **Dijkstra** on your own, from source **A**. Fill a standard table each time you extract the next vertex u and relax outgoing edges.

Dijkstra Trace Table (solution):

Step	Extracted u	$d[A]$	$d[B]$	$d[C]$	$d[D]$	$d[E]$	Check edges / Notes
Init	–	0	∞	∞	∞	∞	Initialize all $d[\cdot]$ to ∞, set $d[A] = 0$.
1	A	0	1	5	∞	∞	Check $A \to B$ (set $d[B] = 1$), relax $A \to C$ (set $d[C] = 5$).
2	B	0	1	2	3	∞	Check $B \to D$ (set $d[D] = 3$); relax $B \to C$ (improve $d[C]$ from 5 to 2).
3	C	0	1	2	3	6	Check $C \to E$ (set $d[E] = 6$).
4	D	0	1	2	3	6	No outgoing edges to check (no change).
5	E	0	1	2	3	6	No outgoing edges to check (final distances).

Shortest weighted path (min total effort) from A to E:

A → B → C → E with total cost $1 + 1 + 4 = 6$

> **Civic Reflection (discuss in groups).** Compared to BFS, which route changed and why? What does your weight choice implicitly value (e.g., high-trust ties, high-bandwidth users, dominant language groups)? Who benefits/hurts?

3

Part C. Pseudocode

BFS(G, start) (for unweighted graphs)

```
for each vertex v in G:
    distance[v] = infinity
    parent[v] = NIL

distance[start] = 0
create an empty queue Q
enqueue(Q, start)

while Q is not empty:
    u = dequeue(Q)
    for each neighbor v of u:
        if distance[v] is infinity:
            distance[v] = distance[u] + 1
            parent[v] = u
            enqueue(Q, v)
```

Dijkstra(G, start) (for weighted graphs with no negative weights)

```
for each vertex v in G:
    distance[v] = infinity
    parent[v] = NIL

distance[start] = 0
create a priority queue PQ (ordered by smallest distance)
insert all vertices into PQ

while PQ is not empty:
    u = vertex in PQ with smallest distance
    remove u from PQ
    for each neighbor v of u:
        if distance[u] + w(u, v) < distance[v]:
            distance[v] = distance[u] + w(u, v)
            parent[v] = u
            update PQ with new distance[v]
```

Part D. Compare & Reflect

1. When would you intentionally choose **BFS** for social systems (equal steps)? Give one concrete case where that equality is desirable.

   Example: A phone tree where every volunteer makes the same kind of short call, and we only care about the fewest number of handoffs, not data plans or language barriers.

2. When must you use **Dijkstra** because equality hides inequity? Name one weight you would add for a fairer outreach or safer information flow.

   Example: Use Dijkstra when some people face higher costs (limited data, language barriers, or risk for speaking out); we could weight edges by time delay, data cost, or safety risk so paths that avoid over-burdened or high-risk users are preferred.

Meta-Reflection:
When all connections are treated equally, we assume that every message spreads with the same ease—this is what Breadth-First Search (BFS) models.

However, in reality, connections often carry different "qualities", such as time delays, language barriers, or varying levels of trust. Using Dijkstra's algorithm and edge weights lets us represent these frictions and study how they affect who receives information first. By analyzing the shortest paths in a weighted graph, we can better understand inequalities in communication, reveal which links make a network more inclusive or exclusive, and design fairer ways for information to reach everyone.