Cammini Minimi
(Shortest Paths)

Alessandro Hill

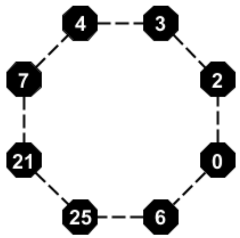
rev. 1.0(AH) – 2024

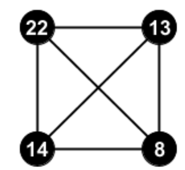
edge = lato (in Grafo non orientato)
arch = archi (in Grafo orientato)

Networks and Notation

Un grafo è connesso se esiste un percorso tra ogni coppia di nodi, ovvero se è possibile passare da qualsiasi nodo a qualsiasi altro nodo del grafo seguendo gli archi. In altre parole, un grafo connesso non ha nodi isolati o parti separate: ogni nodo è collegato indirettamente o direttamente agli altri nodi.

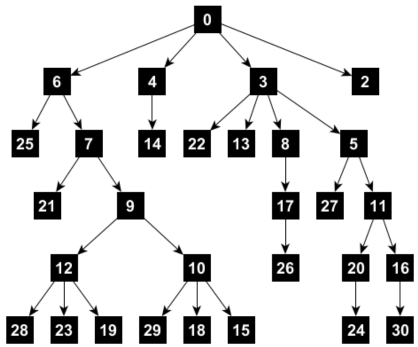
A **network** (or **graph**) consists of a set of **vertices** (or **nodes)** and a set of **edges** that connect selected pairs of nodes.





graph

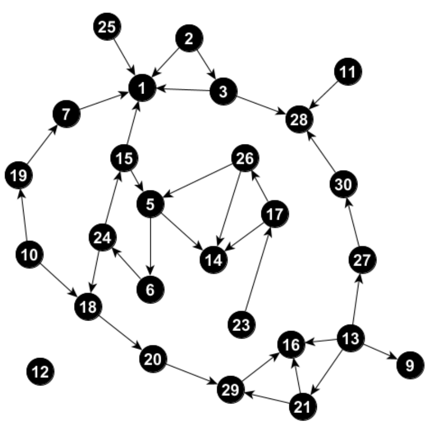
A **directed network** (or **oriented network**) consists of a set of **nodes** and a set of **arcs** (directed edges) that connect selected pairs of nodes.



The **neighbors** of a node are the nodes that are directly connected to the node (i.e., via edges or arcs).

The **degree** of a node is its number of neighbors. For directed networks: **in-degree** and **out-degree**.

vicini in entrata (al nodo)   vicini in uscita (dal nodo)

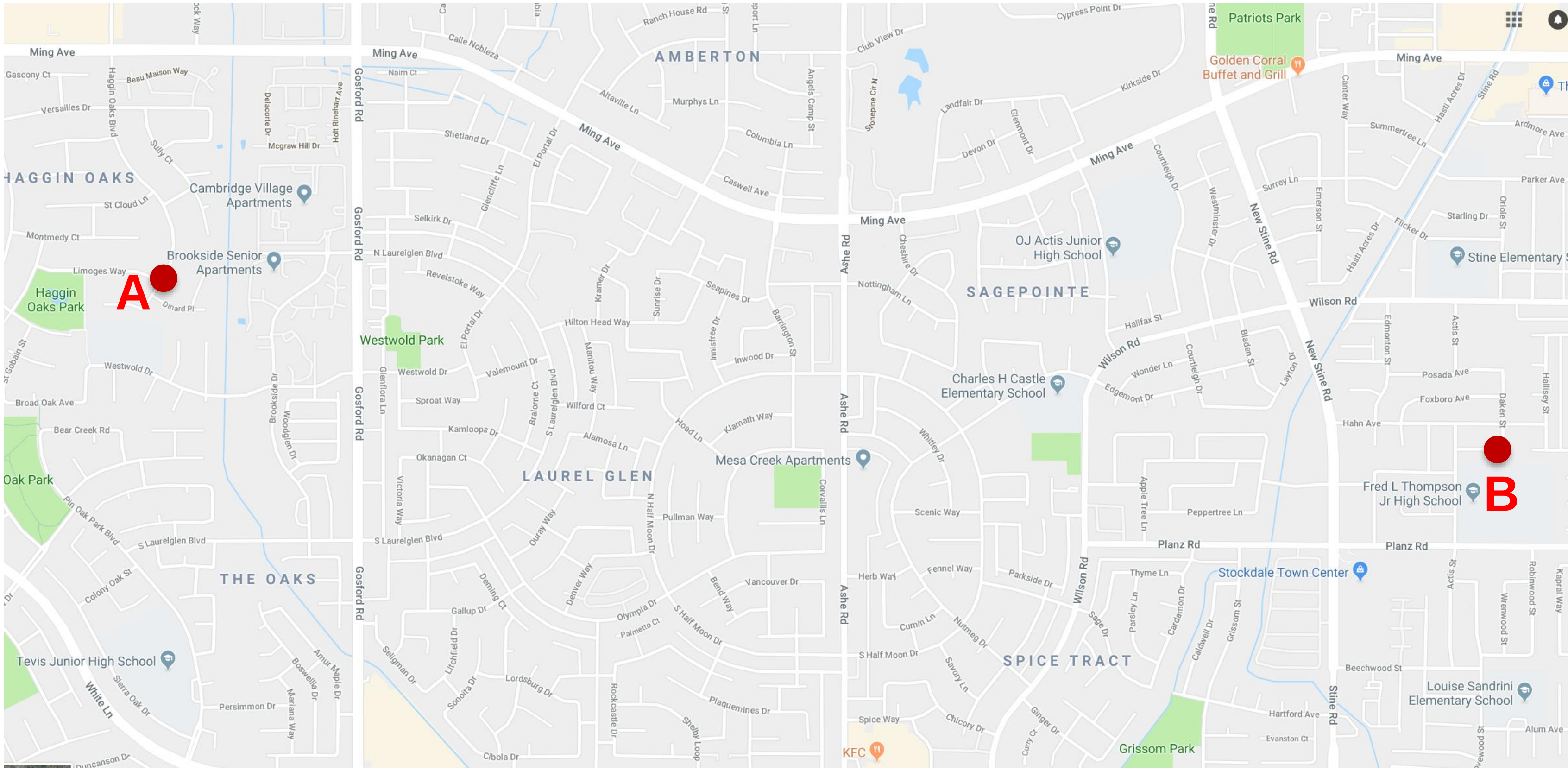


Nodes, edges and arcs may have **weights** (costs or profits) associated with them.

Dijkstra é l'Algoritmo per calcolare Percorso Minimo da una Sorgente (si puó applicare sia su Grafi orientati che non con pesi non negativi)

The Shortest Path Problem

What is the shortest path from **A** to **B**?



- **Nodes:** Intersections, U-turn locations
- **Arcs:** Street segments between nodes
- **Possible arc weights:** Distance, travel time, etc.

Applications: Route planning, transportation, machine control, etc.

Bike, drive, walk, run?

in-neighbors = vicini in entrata nel nodo ; out-neighbors = vicini in uscita a cui il nodo punta

The Shortest Path Problem

IP Model (shortest path from A to B)

Binary arc variables for each arc that could be part of a shortest path:

$$x_{i,j} = \begin{cases} 1 & \text{if arc } (i,j) \text{ will be used on the shortest path,} \\ 0 & \text{otherwise.} \end{cases}$$

Objective:

Funzione Obiettivo

Minimize $\sum_{(i,j) \in A} w_{i,j} x_{i,j}$

Vincoli

Force source node out-flow: Flusso uscente dal nodo di origine A: la somma delle variabili binarie per gli archi che escono da A deve essere uguale a 1, garantendo che il cammino inizi dal nodo A.

$x_{A,j_1} + \ldots + x_{A,j_k} = 1$ for out-neighbors $j_1, \ldots, j_k$ of A.

Vincolo solo per nodo origine

No source node in-flow: Nessun flusso entrante nel nodo di origine A: non deve esserci flusso che entra in A, quindi la somma delle variabili binarie per gli archi che arrivano ad A deve essere zero.

$x_{j_1,A} + \ldots + x_{j_k,A} = 0$ for in-neighbors $j_1, \ldots, j_k$ of A.

Conserve flow through intermediate nodes:

Conservazione del flusso nei nodi intermedi: per ciascun nodo intermedio v, la somma del flusso in entrata deve essere uguale alla somma del flusso in uscita, garantendo così la continuità del percorso.

$x_{i_1,v} + \ldots + x_{i_k,v} = x_{v,j_1} + \ldots + x_{v,j_k}$

for in-neighbors $i_1, \ldots, i_k$, out-neighbors $j_1, \ldots, j_k$ and each intermediate node v.

Force sink node in-flow: Flusso entrante nel nodo di destinazione B: la somma delle variabili binarie per gli archi che arrivano a B deve essere uguale a 1, garantendo che il cammino termini nel nodo B.

$x_{i_1,B} + \ldots + x_{i_k,B} = 1$ for in-neighbors $i_1, \ldots, i_k$ of B.

The Shortest Path Problem

Dijkstra's Algorithm, 1956 (shortest path from A to all other nodes)

L'algoritmo di Dijkstra è una tecnica fondamentale per trovare il percorso più breve da un nodo iniziale (chiamato sorgente) verso tutti gli altri nodi di un grafo orientato o non orientato, con pesi non negativi sugli archi.

1. Initialize node distances: $d(i) = \infty$ for each node I
2. Set distance to zero for start node: $d(A) = 0$
3. Mark all nodes as unvisited
4. Pick unvisited node i that has minimum distance
5. For each unvisited node j that can be reached from i:
   - Distance update $d(j) = \min(d(j) , d(i) + w(i,j))$
   - If distance was updated, set i as predecessor of j: PRED(j) = i
   - Mark i as visited
6. Go to 4 unless all nodes are visited
7. Return shortest path tree encoded in PRED

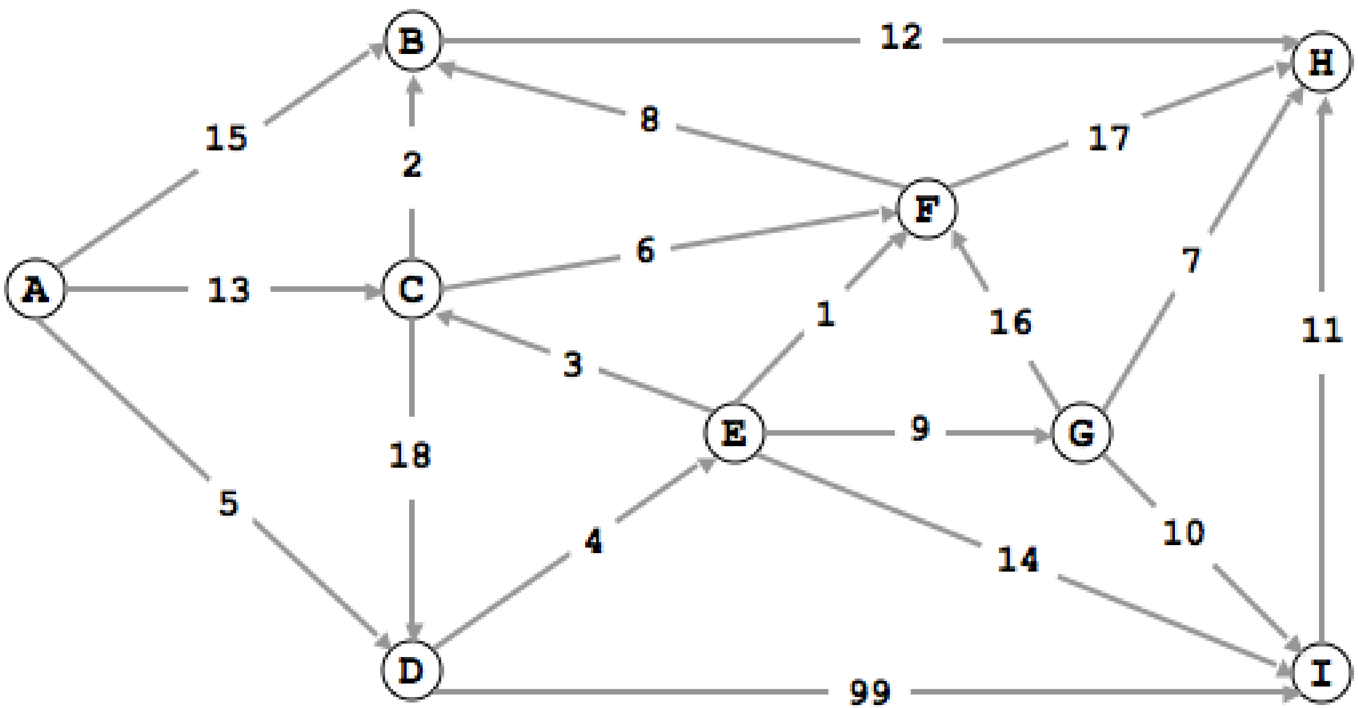
- Inizializzazione delle distanze: Si assegna a ogni nodo una distanza iniziale infinita per indicare che non è ancora stato raggiunto. Al nodo di partenza, invece, si assegna una distanza di 0, perché non c'è alcuna distanza da percorrere per "raggiungere se stesso".
- Marcare tutti i nodi come non visitati: Questo permette di distinguere i nodi che sono stati processati da quelli che devono ancora essere esaminati.
- Scelta del nodo non visitato con distanza minima: Si sceglie tra i nodi non visitati quello con la distanza minima dalla sorgente (cioè, il più vicino al momento). Questo passo è fondamentale per assicurare che il percorso più breve verso ciascun nodo venga calcolato correttamente.

Aggiornamento delle distanze dei vicini: Per ogni nodo adiacente j (cioè, raggiungibile tramite un arco) del nodo i corrente, si verifica se esiste un percorso più breve per raggiungerlo passando attraverso i. Questo aggiornamento è dato dalla formula: $d(j)=\min(d(j),d(i)+w(i,j))$ dove $w(i,j)$ è il peso dell'arco tra i e j. Se $d(j)$ viene aggiornato, significa che il percorso più breve trovato passa per i, quindi i diventa il predecessore di j. Una volta elaborati i vicini di i, questo nodo viene marcato come "visitato", assicurando che non verrà più considerato.

- Iterazione: L'algoritmo ritorna al passo 4 fino a quando tutti i nodi sono stati visitati.
- Risultato: Al termine, la struttura PRED contiene il predecessore di ogni nodo nel percorso più breve, permettendo di ricostruire il cammino. I valori in d rappresentano la distanza minima dalla sorgente a ciascun nodo.

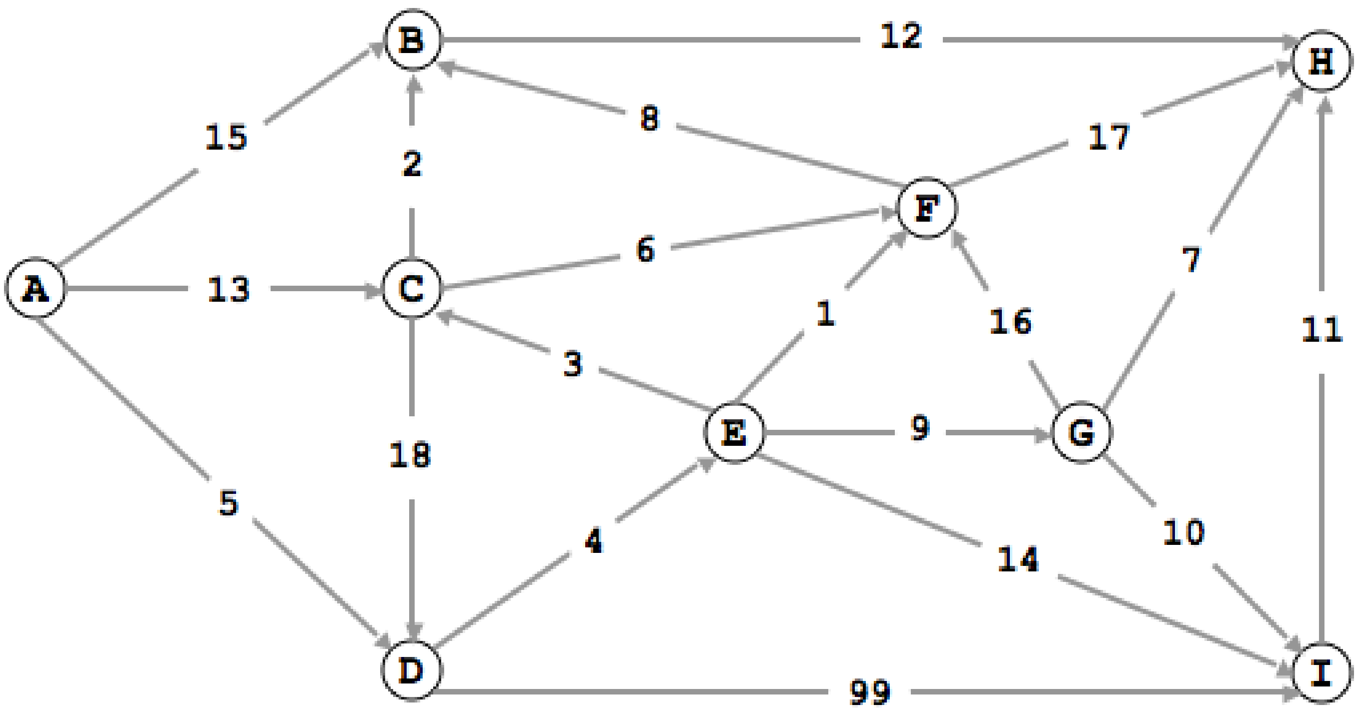
The Shortest Path Problem

1) What is the shortest path from node A to node H in the network given below?



The Shortest Path Problem

2) What is the shortest path from node A to node H in the network given below?

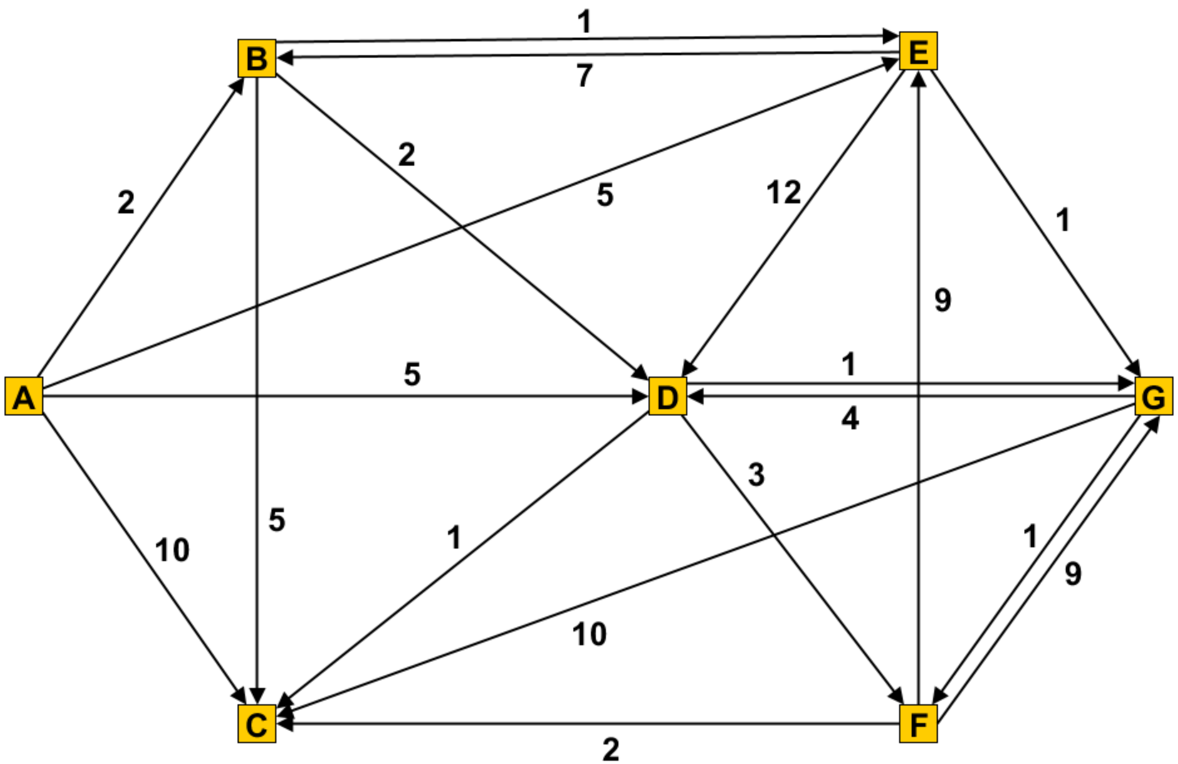


The Shortest Path Problem

1) Find shortest paths using Dijkstra’s algorithm from

a) node A

b) node F



2) Build an IP and find the shortest paths from

c) node A to node G

d) node C to node E