# NetworkX – Riassunto schematico su Grafi, Cammini e Algoritmi

## 1. Cammini (Paths)

- Un cammino è una sequenza di vertici collegati da archi.
- Peso di un cammino = somma dei pesi degli archi.
- Cammino minimo tra u e v: cammino con peso minimo.
- d(u,v) = peso del cammino minimo; d(u,v)=∞ se v non è raggiungibile.

## 2. Problemi di Shortest Path

- Single-Source Shortest Path (SSSP): da una sorgente a tutti.
- All-Pairs Shortest Path (APSP): tra tutte le coppie.
- Si può voler trovare: solo il peso minimo o anche il cammino.
- Rappresentazione: distanza d[v] + predecessore p[v].

## 3. Relaxation

- Tecnica base degli algoritmi di shortest path.
- Se d[v] > d[u] + w(u,v), allora aggiorna d[v] e p[v].
- Garantisce stime sempre ≥ del valore reale.
- Quando d[v] = valore ottimo, non cambia più.

## 4. Algoritmi di Shortest Path

- BFS: grafi non pesati, O(V+E).
- Dijkstra: pesi ≥ 0, greedy, O(E + V log V).
- Bellman-Ford: ammette pesi negativi, rileva cicli negativi, O(V·E).
- Floyd-Warshall: APSP, $O(V^3)$.

## 5. Implementazioni NetworkX – Shortest Path

- nx.shortest_path(G,u,v): cammino minimo (non pesato).
- nx.dijkstra_path(G,u,v,weight): cammino minimo pesato.
- nx.single_source_dijkstra(G,s): distanze + cammini.
- nx.floyd_warshall(G): dizionario distanze APSP.
- nx.all_pairs_bellman_ford_path(G): cammini APSP.

## 6. BFS – Breadth First Search

- Visita per livelli a partire dalla sorgente.
- Usa una coda (FIFO).

- Produce un BFS tree.
- Calcola cammini minimi nei grafi non pesati.

## 7. DFS – Depth First Search

- Visita in profondità.
- Usa stack o ricorsione.
- Utile per: componenti, cicli, DAG, topological sort.
- Produce un DFS tree.

## 8. BFS vs DFS

- BFS: per livelli, shortest path non pesato.
- DFS: struttura del grafo.
- Entrambi hanno complessità O(V+E).

## 9. Cicli nei grafi

- Ciclo: cammino chiuso.
- Ciclo Euleriano: usa ogni arco una sola volta.
- Ciclo Hamiltoniano: visita ogni vertice una sola volta.
- TSP: ricerca del ciclo Hamiltoniano minimo (NP-completo).

## 10. Condizioni Euleriane

- Grafo connesso.
- Ciclo Euleriano ⇔ tutti i vertici hanno grado pari.
- Cammino Euleriano ⇔ al più due vertici di grado dispari.

## 11. NetworkX – Cicli

- nx.is_eulerian(G): verifica ciclo euleriano.
- nx.eulerian_circuit(G): restituisce il ciclo.
- Per Hamilton: nessun algoritmo efficiente generale.

## 12. NetworkX – Strutture Dati

- Nodi: qualsiasi oggetto hashable.
- Archi: tuple con attributi.
- Struttura: dizionario di dizionari.
- Supporto per Graph, DiGraph, MultiGraph.

## 13. Operazioni comuni

- g[u][v]: attributi arco.

- for n in g: itera sui nodi.
- g.nodes(), g.edges().
- successors(), predecessors() per grafi diretti.

## 14. Traversal in NetworkX

- nx.bfs_tree, nx.bfs_edges.
- nx.dfs_tree, nx.dfs_edges.
- Traversal = visita sistematica del grafo.