Algoritmi + Complexitate

Curs 1+2

1. DFS -> O(N+M) <= cycle detection in un/directed graph
2. BFS -> O(N+M) <= cycle detection in undirected graph
3. Havel-Hakimi (=is\_graph) -> O(N2logN) pt ca sort-ul e in O(NlogN) pt fiecare iteratie din n (secv de n grade)
4. Sortare topologica pt DAG -> O(N+M)
5. Kosaraju (fol sort top) -> O(N+M) <= CTC
6. Tarjan -> O(N+M) <= CTC

Curs 3

1. Teorema celor 6 Culori -> O(N)
2. Teorema Konig (e planar?) -> O()
3. T. lui Kuratowski (g planare) -> O()
4. Teorema lui Euler

Curs 5 APCM = MST (minimum spanning tree)

1. Union-Find -> O(logn) <= path compression / cycle detection
2. APCM: Kruskal -> O(m log n + n2) sau cu UF -> O(m log n) <= cycle detection
3. APCM: Prim -> O(n2) sau cu min-heap -> O(m log n)
4. APCM: Boruvka -> O(m log n) <= pt minimum spanning forest

Curs 6+7

1. Drum min in DAG -> O(m+n) <= fol sort top
2. Dijkstra -> O(n2) sau cu min-heap -> O(m log n)
3. Bellman-Ford -> O(mn)
4. Floyd - Warshall -> O(n3) <= inchiderea tranzitiva a unui graf = CTC
5. Roy – Warshall / [negative cycle detection](https://www.geeksforgeeks.org/detecting-negative-cycle-using-floyd-warshall/)

Curs 8

1. Ford-Fulkerson generic -> O(nmC) unde C = max\_flow <- flux/cuplaj maxim
2. Edmonds - Karp (FF cu BFS) -> O(nm2)

Curs 9

1. Hierholzer (ciclu eulerian) -> O(m) <- ciclu eulerian (trece prin fiecare muchie o

singura data) <=> toate nodurile au grad par

1. Ciclu Hamilton de cost min -> O(m\*2n) <- progr dinamica cu memoizare

<- / alg de aprox greedy cu APCM<=OPT<=2\*APCM

cycle = path (drum) that starts and ends with the same vertex

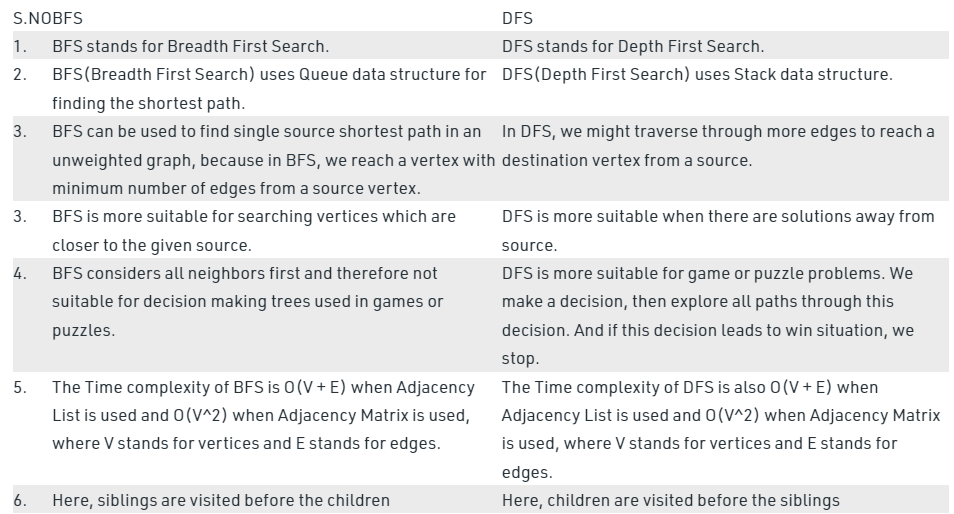
**BFS:**

- [time complexity](https://en.wikipedia.org/wiki/Time_complexity): {\displaystyle O(|V|+|E|)}O(V+E), since every vertex and every edge will be explored in the worst case.

Obs: O(E) {\displaystyle |V|}may vary between {\displaystyle O(1)}O()O(1) and {\displaystyle O(|V|^{2})}O(V2), depending on how sparse (= imprastiat) the input graph is.

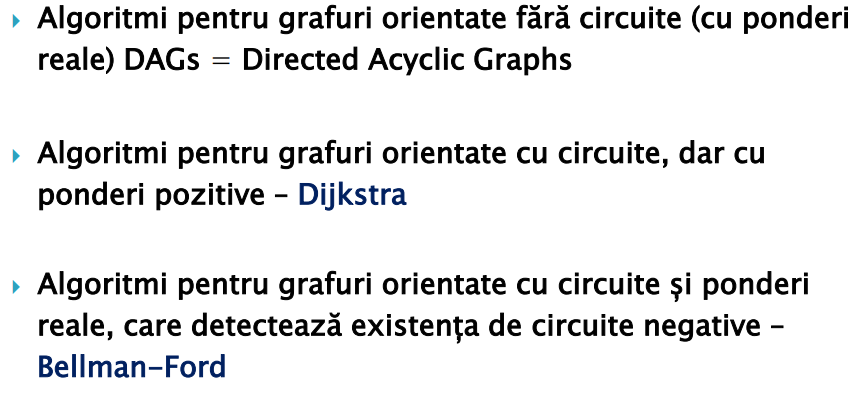
- [space complexity](https://en.wikipedia.org/wiki/Space_complexity): O(V), when the number of vertices in the graph is known ahead of time, and additional data structures are used to determine which vertices have already been added to the queue. This is in addition to the space required for the graph itself, which may vary depending on the [graph representation](https://en.wikipedia.org/wiki/Graph_(abstract_data_type)) used by an implementation of the algorithm.

**BFS versus DFS**



Drumuri minime in grafuri orientate ponderate (=cu costuri pe muchii = w[])

* intre doua noduri date s si v -> Dijkstra
* intre un nod sursa si toate celelalte noduri



* intre oricare doua noduri -> Dijkstra / Floyd–Warshall

Obs1: daca graful nu e ponderat, pb se rez cu BFS din vf sursa

Obs2: daca la Dijkstra toate costurile sunt =, alg e echiv cu BFS