Part I - Graph Problems

Overview

- Depth-First Search (DFS) ⟶ connectivity;
- Dijkstra ⟶ shortest path;
- Prim and Kruskal ⟶ MST;
- Ford-Fulkerson ⟶ maximum flow;
- Maximum Matching ⟶ matchings.

Maximum Matching

Algorithm 1: MAXIMUM MATCHING

Input: undirected graph $G = (V, E)$

Output: maximum matching M

```
set M = ∅
while there exists M-augmenting path in G do
    choose M-augmenting path P
    set M = (M \ E(P)) ∪ (E(P) \ M)
return M
```

Prim's

Algorithm 2: PRIM'S ALGORITHM

Input: undirected, connected graph G, weights $c\colon E(G) \to \mathbb{R}$

Output: spanning tree T of minimum weight

```
choose v ∈ V(G)
set T := ({v}, ∅)
while V(T) ≠ V(G) do
    choose an edge e ∈ δ_G(V(T)) of minimum weight
    set T := T + e
return T
```

Dijkstra's

Algorithm 3: DIJKSTRA'S ALGORITHM

Input: undirected, connected graph G, weights $c\colon E(G) \to \mathbb{R}$, nodes V, source s

```
d_v distance to reach node v
p_v node predecessor to node v
Q ← ∅ set of "unkown distance" nodes.
for each node v in V do
    d_v ← ∞
    p_v ← FALSE
    add v in Q
d_s ← 0 while Q ≠ ∅ do
    u ← node in Q with min d_u
    remove u from Q
    for each neighbor v of u still in Q do
        d ← d_u + c_uv
        if alt < d_v then
            d_v ← alt
            p_v ← u
```

Kruskal's

Algorithm 4: KRUSKAL'S ALGORITHM

Input: undirected, connected graph G, weights $c\colon E(G) \to \mathbb{R}$

Output: spanning tree T of minimum weight

```
sort edges such that c(e_1) ≤ c(e_2) ≤ ... ≤ c(e_m)
set T := (V(G), ∅)
for i := 1 to m do
    if T + e_i contains no cycle then
        set T := T + e_i
return T
```

Ford-Fulkerson's

Algorithm 5: FORD-FULKERSON ALGORITHM

Input: digraph $G = (V, E)$, capacities $u\colon E \to \mathbb{Z}_+$, $s, t, \in V$

Output: maximal s-t-flow f

```
set f(e) = 0 for all e ∈ E
while there exists f-augmenting path in G_f do
    choose f-augmenting path P
    set Δ_f(P) = min_{a∈E(P)} u_f(a)
    augment f along P by Δ_f(P)
    update G_f
return f
```

DFS

Algorithm 6: DEPTH FIRST SEARCH (DFS)

Input: undirected graph G, vertex $s \in V(G)$

Output: tree $(R, T) \subseteq G$, R reachable from s

```
1 set R := {s}, Q := {s} and T = ∅;
2 if Q = ∅ then return R, T;
3 else v := last vertex added to Q;
4 choose w ∈ V(G) \ R with {v, w} ∈ E(G);
5 if there is no such w then
6     set Q := Q \ {v} and go to 2
7 set R := R ∪ {w}, Q := Q ∪ {w}, T := T ∪ {{v, w}}, go to 2;
```

Fernando Dias
fernando.dias@aalto.fi
Combinatorial Optimization
Aalto University