IB002 cheat sheet
Jaro 2021

```
1  function ShortBFS((V, E), s)
2      foreach u ∈ V \ {s} do
3          u.visited = False
4      s.visited = True
5      queue = empty queue
6      Enqueque (queue, s)
7      while queue is not empty do
8          u = Dequeque(queue)
9          foreach v ∈ u.successors do
10             if not v.visited then
11                 v.visited = True
12                 Enqueque (queue, v)
```

```
1  function FullBFS((V, E), s)
2      foreach u ∈ V \ {s} do
3          u.color = white
4          u.distance = ∞
5          u.π = None
6      s.color = gray
7      s.distance = 0
8      s.π = None
9      queue = empty queue
10     Enqueque (queue, s)
11     while queue is not empty do
12         u = Dequeque(queue)
13         foreach v ∈ u.successors do
14             if v.color == white then
15                 v.color = gray
16                 v.distance = u.distance + 1
17                 v.π = u
18                 Enqueque (queue, v)
19         u.color = black
```

```
1  function DFS((V, E))
2      foreach u ∈ V do
3          u.color = white
4          u.π = None
5      time = 0
6      foreach u ∈ V do
7          if u.color == white then
8              time = FullDfsVisit ((V, E), u, time)
```

```
1  function FullDfsVisit((V, E), u, time)
2      time += 1
3      u.discovery = time
4      u.color = gray
5      foreach v ∈ u.successors do
6          if v.color == white then
7              v.π = u
8              time = FullDfsVisit ((V, E), v, time)
9      u.color = black
10     time += 1
11     u.finish = time
12     return time
```

```
1  function ShortDFS((V, E))
2      foreach u ∈ V do
3          u.visited = False
4      foreach u ∈ V do
5          if not u.visited then
6              ShortDfsVisit ((V, E), u)
```

```
1  function ShortDfsVisit((V, E), u)
2      u.visited = True
3      foreach v ∈ u.successors do
4          if not v.visited then
5              ShortDfsVisit ((V, E), v)
```

```
function InitSssp((V, E), s)
    foreach v ∈ V do
        v.d = ∞
        v.π = None
    s.d = 0
```

```
function Relax(u, v, w)
    v.d = u.d + w(u, v)
    v.π = u
```

```
function Bellman-Ford((V, E), w, s)
    InitSssp ((V, E), s)
    for i = 1 to |V| − 1 do
        foreach (u, v) ∈ E do
            if v.d > u.d + w(u, v) then
                Relax (u, v, w)
    foreach (u, v) ∈ E do
        if v.d > u.d + w(u, v) then
            return False
    return True
```

```
function Dijkstra((V, E), w, s)
    InitSssp ((V, E), s)
    Q = ∅  //Q je prioritní frona
    Insert (Q, s, s.d)
    while Q is not empty do
        u = ExtractMin (Q)
        foreach v ∈ u.successors do
            if v.d == ∞ then
                Insert (Q, v, v.d)
            if v.d > u.d + w(u, v) then
                Relax (u, v, w)
                DecreaseKey (Q, v, v.d)
```