Algorithmique et Programmation II

Recherche de plus courts chemins

Considérons un graphe orienté pondéré $G = (V, E)$, de fonction de poids w, et une origine s.

1. Implémenter et comparer les deux versions de l'algorithme de Dijkstra, sans et avec tas (cf. Algorithme 1 et Algorithme 2).
2. Implémenter l'algorithme de Bellman-Ford (cf. Algorithme 3). Quel est l'avantage de l'algorithme de Bellman-Ford par rapport à celui de Dijkstra ?
3. Appliquer les algorithmes implémentés sur les instances accessibles au lien suivant : http://www.dis.uniroma1.it/challenge9/download.shtml.

Algorithm 1 : Dijkstra simple (G, w, s)

```
1:  d(u) ← ∞, ∀u ∈ V
2:  state(u) ← unreached, ∀u ∈ V
3:  d(s) ← 0
4:  queue.insert(s)
5:  while (queue.empty() == false) do
6:    u ← v, where v ∈ queue with d(v) minimal
7:    state(u) ← reached
8:    queue.remove(u)
9:    for all outgoing edges (u, v) of u do
10:     if d(u) + w(u, v) < d(v) then
11:       d(v) ← d(u) + w(u, v)
12:       if state(v) == unreached then
13:         queue.insert(v)
14:         state(v) ← reached
15:       end if
16:     end if
17:   end for
18: end while
```

Algorithm 2 : Dijkstra avec tas (G, w, s)

```
1:  d(u) ← ∞, ∀u ∈ V
2:  state(u) ← unreached, ∀u ∈ V
3:  d(s) ← 0
4:  priority_queue.insert(s)
5:  while (priority_queue.empty() == false) do
6:    u ← priority_queue.extractMin()
7:    state(u) ← reached
8:    priority_queue.remove(u)
9:    for all outgoing edges (u, v) of u do
10:     if d(u) + w(u, v) < d(v) then
11:       d(v) ← d(u) + w(u, v)
12:       if state(v) == unreached then
13:         priority_queue.insert(v, d(v))
14:         state(v) ← reached
15:       else
16:         priority_queue.decreaseKey(v, d(v))
17:       end if
18:     end if
19:   end for
20: end while
```

Algorithm 3 : Bellman-Ford $G(V, E, s)$

```
for all v ∈ V do
  if v == s then
    d(v) := 0
  else
    d(v) ← ∞
  end if
end for
for all i from 1 to |V| − 1 do
  for all edge (u, v) ∈ E do
    if d(u) + w(u, v) < d(v) then
      d(v) ← d(u) + w(u, v)
    end if
  end for
end for
for all edge (u, v) ∈ E do
  if d(u) + w(u, v) < d(v) then
    return false
  else
    return true
  end if
end for
```