В алгоритмите по долу ползваме приоритетна опашка (по минимум), в която пъхаме тройки от вида $\langle key, u, v\rangle$, където key е стойност, според която нареждаме тройките в опашката, u и v са двойка върхове в граф (може някое да е NIL).

MST Даден е неориентиран граф $G(V, E)$ с теглова функция $w : E \to \mathbb{R}$.

Търсим минималното покриващо дърво за G.

```
MST_Prim_PQ(G(V,E), w, s)
for v ∈ V
    marked[v] ← False
PQ.Init(minHeap)
PQ.Insert(⟨0, NIL, s⟩) (∗ Фиктивно ребро до s ∗)
while PQ ≠ ∅ do
    ⟨dist, pred, u⟩ ← PQ.GetMin
    if not marked[u]
        marked[u] ← True
        d[u] ← dist
        π[u] ← pred
        for v ∈ Adj(u)
            PQ.Insert(⟨ w(u, v) , u, v⟩)
return d, π (∗ Тегла на ребрата и кореново дърво, представящи MST ∗)
```

Най-кратки пътища Даден е ориентиран граф $G(V, E)$ с теглова функция $w : E \to \mathbb{R}^+$.

Търсим най-кратките пътища от s до останалите върхове на G.

```
Dijkstra_PQ(G(V,E), w, s)
for v ∈ V
    marked[v] ← False
PQ.Init(minHeap)
PQ.Insert(⟨0, NIL, s⟩) (∗ Фиктивно ребро до s ∗)
while PQ ≠ ∅ do
    ⟨dist, pred, u⟩ ← PQ.GetMin
    if not marked[u]
        marked[u] ← True
        d[u] ← dist
        π[u] ← pred
        for v ∈ Adj(u)
            PQ.Insert(⟨ dist + w(u, v) , u, v⟩)
return d, π (∗ Дължини на минималните пътища и кореново дърво за тях ∗)
```