```
prim(GRAPH G, NODE r, int[ ] p)
    // r:  nodo dalla quale parto
    // p:  vettore dei padri

    PRIORITYQUEUE Q ← MinPriorityQueue
    PRIORITYITEM[] POS ← new PRIORITYITEM[1...G.n]

    // inserisco i nodi nella coda, memorizzando la loro posizione
(1) foreach u ∈ G.V() − {r} do
        POS[u] ← Q.inserisci(u, +∞)

    // Inserisco il "nodo di partenza"
    POS[r] ← Q.inserisci(r, 0)
    p[r] ← 0  // convenzione per indicare che non ha padre

(2) while not Q.isEmpty do // non ci sono più nodi
        NODE u ← Q.deleteMin  // cancello e restituisco il nodo
        POS[u] ← nil // non considero più quel nodo

        // per ciascun nodo adiacente a quello considerato
(3)     foreach v ∈ G.adj(u) do
            if POS[v] ≠ nil and w(u,v) < POS[v].priority then
                // POS[v] ≠ nil:  è già stato visitato
                // w(u,v) < POS[v].priority:
                Q.decrease(POS [v], w(u,v))  // commento
                p[v] ← u  // commento
```