

```

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]

    (1) // inserisco i nodi nella coda, memorizzando la loro posizione
        per ciascun  $u \in G.V() - \{r\}$  fai
            pos[u] ← Q.inserisci( $u, +\infty$ )
    // Inserisco il "nodo di partenza"
    pos[r] ← Q.inserisci(r, 0)
    p[r] ← 0 // convenzione per indicare che non ha padre

    (2) finché not Q.isEmpty fai // 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
        per ciascun  $v \in G.adj(u)$  fai
            se  $pos[v] \neq \text{nil}$  and  $w(u,v) < pos[v].priority$  allora
                //  $pos[v] \neq \text{nil}$ : è già stato visitato
                //  $w(u,v) < pos[v].priority$ :
                Q.decrease( $Pos[v]$ ,  $w(u,v)$ ) // commento
                p[v] ← u // commento

```