

(int, int) CamminiMinimi(GRAPH  $G$ , NODE  $s$ )

```
// Inizializzazione dei vettori
int[]  $d \leftarrow$  new int[1... $G.n$ ] // distanze dalla sorgente
int[]  $T \leftarrow$  new int[1... $G.n$ ] // vettore dei padri
bool[]  $b \leftarrow$  new bool[1... $G.n$ ] // per sapere in tempo costante se  $u \in S$ 

// Inizializzo tutti i nodi tranne la sorgente
foreach  $u \in G.V - \{s\}$  do
     $T[u] \leftarrow$  nil // non hanno padri
     $d[u] \leftarrow +\infty$  // non li ho ancora raggiunti
     $b[u] \leftarrow$  false // non appartengono ancora all'insieme

// Inizializzo la sorgente
 $T[s] \leftarrow$  nil // non ha padre
 $d[s] \leftarrow 0$  // per convenzione
 $b[s] \leftarrow$  true // appartiene all'insieme

(1) STRUTTURADATI  $S \leftarrow$  StrutturaDati
     $S.aggiungi(s)$ 

(2) while not  $S.isEmpty$  do
    int  $u \leftarrow S.estrai$  // estraggo un nodo
     $b[u] \leftarrow$  false // non è più contenuto nella struttura dati

    foreach  $v \in G.adj(u)$  do // per tutti i vicini
        (3) if  $d[u] + G.w(u, v) < d[v]$  then // se migliora la stima
            if not  $b[v]$  then // se non fa già parte dell'insieme
                 $S.aggiungi(v)$  // aggiungilo
                 $b[v] \leftarrow$  true // fa parte dell'insieme
            else
                (4) // Azione da intraprendere nel caso  $v$  sia già presente in  $S$ 

                // aggiorni i vettori
                 $T[v] \leftarrow u$ 
                 $d[v] \leftarrow d[u] + G.w(u, v)$ 

return ( $T, d$ )
```