

### Algoritmo 0.1 – Algoritmo generico

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 un nodo appartiene
    alla struttura dati

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

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

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

finché not  $S.isEmpty$  fai
(2)     int  $u \leftarrow S.estrai$  // estraggo un nodo
         $b[u] \leftarrow$  falso // non è più contenuto nella struttura dati
        per ciascun  $v \in G.adj(u)$  fai // per tutti i vicini
            se  $d[u] + G.w(u,v) < d[v]$  allora // se migliora la stima
                se not  $b[v]$  allora // se non fa già parte dell'insieme
                    (3)      $S.aggiungi(v)$  // aggiungilo
                             $b[v] \leftarrow$  vero // fa parte dell'insieme
                altrimenti
                    (4)     // Azione da intraprendere nel caso  $v$  sia già presente in  $S$ 

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

        ritorna ( $T, d$ )
```