**algoritmo** extractMin (**Set** Q di **Vertici**) → **Vertice**

**double** min\_dist := Infinito

**Vertice** min\_vu := null

**foreach** Vertice v in Q **do**

**if** (v.percorso.dist\_totale < min\_dist) **then**

min\_dist := v.percorso.dist\_totale

min\_vu := v

**endif**

**endfor**

**if** (min\_vu != null) **then**

rimuovi min\_vu da Q

**endif**

**return** min\_vu

**algoritmo** getPercorsiMinimi (**Grafo** G = (V, E), **String** idAcquirente) → **List** di **Percorsi**

**Vertice** acquirente := Vertice in V con id uguale a idAcquirente

**foreach** vertice v in V **do**

v.percorso.nodi := ∅

v.percorso.distanze := ∅

v.percorso.passi := 0

v.percorso.dist\_totale := Infinito

**endfor**

acquirente.percorso.dist\_totale := 0

**List** di **Percorso** percorsi := ∅

**Set** di **Utente** Q := ∅

Aggiungi a Q tutti i vertici del grafo G

**Vertice** vu

**while** (Q != ∅) **do**

vu := extraxtMin(Q)

**if** (vu == null) **then**

**break**;

**endif**

aggiungi vu.percosro in coda a percorsi

**if** (vu.percorso.passi >= 3) **then**

**continue**;

**endif**

**foreach** vertice vua adiacente a vu **do**

**if** (vua.percorso.dist\_totale > vu.percorso.dist\_totale  
 + peso arco (vu,vua) **then**

vua.percorso.nodi :=   
vu.percorso.nodi U {vua.id}

vua.percorso.distanze :=   
vu.percorso.distanze U {peso arco (vu,vua)}

vua.percorso.passi :=   
vu.percorso.passi + 1

vua.percorso.dist\_totale := vu.percorso.dist\_totale + peso arco (vu,vua)

**endif**

**endfor**

**endwhile**

rimuovi il primo elemento di percorsi

**return** percorsi

**algoritmo** getLibriAcquistabili (**Grafo** G = (V, E), **String** idAcquirente) → **List** di **SpecificheAcquisto**

**List** di **Percorso** percorsi := getPercorsiMinimi(G, idAcquirente)

**List** di **SpecificheAcquisto** result := ∅

**Utente** u := null

**List** di **String** utenti\_percorso := null

**List** di **int** token\_percorso := null

**foreach** Percorso p in percorsi **do**

u := Utente con id pari alla prima stringa in p.nodi

utenti\_percorso := p.nodi

token\_percorso := ∅

**foreach** double d in p.distanze **do**

calcola il valore in token relativo alla distanza e aggiungilo a token\_percorso

**endfor**

**foreach** Libro l in u.libri **do**

crea un nuovo SpecificheAcquisto con libro := l,  
utenti := utenti\_percorso, tokens := token\_percorso

calcola il valore in token del libro e sommalo al primo elemento di tokens di SpecificheAcquisto

aggiungi l’oggetto così creato a result

**endfor**

**endfor**

**return** result;