Componenti del gruppo:

899826 Roncoroni Daniele

869525 Marco Tantardini

Descrizione funzioni

(is-graph (graph-id))

Questa funzione ritorna il graph-id stesso se questo grafo è già stato creato, oppure NIL se no.

(delete-graph (graph-id))

Rimuove l’intero grafo dal sistema (vertici archi etc); ovvero rimuove tutte le istanze

presenti nei data base (ovvero nelle hash-tables) del sistema.

(new-graph (graph-id))

Questa funzione genera un nuovo grafo e lo inserisce nel data base (ovvero nella hash-table)

dei grafi.

(new-vertex (graph-id vertex-id))

Aggiunge un nuovo vertice *vertex-id* al grafo *graph-id*.

(graph-vertices (graph-id))

Questa funzione torna una lista di vertici del grafo.

(new-edge (graph-id vertex-id1 vertex-id2 &optional weight))

Questa funzione aggiunge un arco del grafo graph-id nella hash-table \*edges\*.

(graph-edges (graph-id))

Questa funzione ritorna una lista una lista di tutti gli archi presenti in graph-id.

(graph-vertex-neighbors (graph-id vertex-id))

Questa funzione ritorna una lista *vertex-rep-list* contenente gli archi

(edge graph-id vertex-id N W),

che portano ai vertici N immediatamente raggiungibili da vertex-id.

(graph-print (graph-id))

Questa funzione stampa alla console dell’interprete Common Lisp una lista dei vertici e degli

archi del grafo graph-id.

;sssp

(sssp-dist (graph-id vertex-id))

Questa funzione, dato un *vertex-id* di un grafo *graph-id* ritorna, durante e dopo l’esecuzione

dell’algoritmo di Dijkstra, la distanza minima *d* del vertice *vertex-id* dalla “sorgente”

(sssp-visited (graph-id vertex-id))

Questo predicato è vero quando *vertex-id* è un vertice di *graph-id* e, durante e dopo

l’esecuzione dell’algoritmo di Dijkstra, *vertex-id* risulta “visitato”

(sssp-previous (graph-id vertex-id))

Questa funzione, durante e dopo l’esecuzione dell’algoritmo di Dijkstra, ritorna il vertice *U* che è il

vertice “precedente” a *V* nel cammino minimo dalla “sorgente” a *V*

(sssp-change-dist (graph-id vertex-id new-dist))

Questa funzione ha solo un effetto collaterale: alla chiave (*graph-id* *V*)

nella hash-table \*distances\* viene associato il valore *new-dist*

(sssp-change-previous (graph-id vertex-id new-previous))

Questa funzione ha solo un effetto collaterale: alla chiave(*graph-id* *V*)

nella hash-table \*previous\* viene associato il valore *U*.

(sssp-dijkstra (graph-id source))

Questa funzione termina con un effetto collaterale. Dopo la sua esecuzione, la hash-table

\*distances\* contiene al suo interno le associazioni (*graph-id* *V*) ⇒ *d* per ogni V

appartenente a graph-id; la hash-table \*previous\* contiene le associazioni (*graph-id* *V*) ⇒ *U*;

infine la hash-table \*visited\* contiene the associazioni *graph-id* *V*) ⇒ {T, NIL}.

(sssp-dijkstra-helper (graph-id))

DANIELE

(update (graph-id vertice array lunghezza))

DANIELE

(update-helper (graph-id vertice-id array punto))

DANIELE

(heap-modify-distance (heap-id key-new key-old))

DANIELE

(find (array size chiave))

DANIELE

(sssp-shortest-path (graph-id source vertex-id))

Questa funzione ritorna una lista di archi che rappresenta il “cammino minimo” da Source a V.

(sssp-shortest-path-helper (graph-id source vertex-id elementi))

DANIELE

(all-edges (graph-id vertex-id1 vertex-id2))

DANIELE

;MinHeap

(new-heap (heap-id capacity))

Questa funzione inserisce un nuovo heap nella hash-table \*heaps\*.

(heap-id (heap-id))

Ritorna l’id dell’heap

(heap-size (heap-id))

Ritorna la dimensione dell’heap

(the-heap (heap-id))

Ritorna il valore dell’heap

(heap-capacity (heap-id))

Metodo ausiliario che ritorna la capacità dell’heap

(heap-delete (heap-id))

Rimuove tutto lo heap indicizzato da *heap-id*. Potete usare la funzione **remhash** per questo

scopo.

(heap-empty (heap-id))

Questo predicato è vero quando lo heap *heap-id* non contiene elementi.

(heap-not-empty (heap-id))

Questo predicato è vero quando lo heap *heap-id* contiene almeno un elemento.

(heap-head (heap-id))

La funzione heap-head ritorna una lista di due elementi dove K è la chiave minima e V il valore

associato.

(heap-print heap-id)

Questa funzione stampa sulla console lo stato interno dello heap heap-id. Questa funzione vi

serve soprattutto per debugging; il formato di questa stampa è libero.

(heap-insert (heap-id key value))

La funzione heap-insert inserisce l’elemento V nello heap heap-id con chiave K.

Naturalmente, lo heap heap-id dovrà essere ristrutturato in modo da mantenere la “heap

property” ad ogni nodo dello heap. Richiama la funzione (order-heap) per ripristinare la Heap-Priority

A sua volta order-heap richiama (switch-el) e (switch-ot) che si occupano

di invertire gli elementi e ristabilire ordine nell'heap

(heap-extract (heap-id))

La funzione heap-extract ritorna la lista con K, V e con K minima; la coppia è rimossa dallo

heap heap-id. Naturalmente, lo heap heap-id dovrà essere ristrutturato in modo da mantenere

la “heap property” ad ogni nodo dello heap. Richiama la funzione (order-el-ex) per ripristinare la Heap-Priority

A sua volta order-el richiama (switch-to-nill) e (switch-ot-ex) che si occupano

di invertire gli elementi e ristabilire ordine nell'heap

(heap-modify-key (heap-id new-key old-key value))

La funzone heap-modify-key sostituisce la chiave OldKey (associata al valore V) con

NewKey. Naturalmente, lo heap heap-id dovrà essere ristrutturato in modo da mantenere la

“heap property” ad ogni nodo dello heap. Per ritrovare la posizione della chiave da sostituire viene utilizzata la funzione (find-position). Ristabilisce successivamente la Heap-Priority con le funzioni

(heap-shift)

(heap-shift-right) --> shift destro dell'heap

(heap-shift-left) --> shift sinistro dell'heap