

Descrizione delle strategie risolutive e delle strutture dati adottate

La struttura dati utilizzata per rappresentare il grafo contiene alcuni campi standard (interi "V", "E", tabella di simboli "st"), la matrice di adiacenza (si è scelta la matrice nonostante possa essere sconveniente da un punto di vista della memoria per grafi sparsi, principalmente per beneficiare dell'accesso diretto nelle operazioni di rimozione degli archi dal grafo, che nei problemi proposti possono ricorrere frequentemente), un vettore di interi "vertex" per il marcaggio dei vertici (0 vertice non fa parte del grafo, 1 vertice fa parte del grafo), e un intero "nV" (per il conteggio dei vertici che in un certo istante fanno effettivamente parte del grafo).

Per la risoluzione del primo problema proposto (k-core) si è seguito un approccio ricorsivo: si verifica se qualche vertice ha grado minore di k, se si ha esito positivo si procede con la sua rimozione (setstando opportunamente la cella corrispondente del vettore "vertex") e con la rimozione degli archi che su di esso insistono, e così via sino a giungere ad un'istanza della chiamata ricorsiva in cui non si effettueranno più rimozioni. Giunti quindi a questo caso terminale si è ottenuto l'insieme di vertici (potenzialmente anche vuoto) appartenenti al k-core, per cui si stampa la soluzione a video. Si chiudono dunque le chiamate ricorsive tramite flag e si ripristinano opportunamente le condizioni iniziali del grafo (si aggiungono i vertici e gli archi precedentemente eliminati).

Per la risoluzione del secondo problema proposto (j-edge-connected) si è seguito invece un approccio simile al powerset con combinazioni semplici: si estraggono tutti gli archi del grafo e si memorizzano in un vettore "val", tramite ciclo esterno si itera con un indice "i" variabile tra 1 e $G \rightarrow E$, si calcolano le combinazioni semplici degli $|E|$ archi a i a i e in ogni caso terminale si verifica se il sottoinsieme di archi calcolato sconnette il grafo (tramite rimozione degli archi e funzione standard GRAPHcc, per poi reinserire gli archi); una volta trovato un insieme di i archi che sconnette il grafo (opportunamente segnalato tramite valore di ritorno della funzione comb_sempl) si verifica se questo insieme ha cardinalità minore, uguale o maggiore di j, da ciò dipenderà il risultato finale ($i < j$ grafo non j-edge-connected, $i \geq j$ grafo j-edge-connected).

Note

Ho creato due file di input, un file grafo2.txt che riproduce il grafo G della traccia e utilizzabile per testare il punto 2 dell'esame, e un file grafo3.txt che riproduce il grafo esempio per il punto 3 dell'esame (che altri non è che il 3-core di G) e con cui si può per l'appunto testare l'algoritmo per la verifica j-edge-connected.