Algoritmi e Strutture Dati (II anno) - a.a. 1996-97

Ultima modifica: 9 Aprile 1997. Commenti a Eugenio Moggi.

Rappresentazione di grafi orientati

Le rappresentazioni piu' comunemente utilizzate per un grafo (orientato) sone quelle a matrice di adiacenza e a liste di adiacenza. Supponiamo fissato l'insieme dei nodi V, allora *astrattamente* un grafo G=(V,E) puo' essere rappresentato

- con un array di insiemi di adiacenza

  ```
  A: array [V] of "sottoinsiemi di V"
  ```

  la relazione di corrispondenza tra grafo e sua rappresentazione e' data da: (u,v) e' un arco sse v e' in A[u].
- con un insieme di coppie ordinate

  ```
  B: "sottoinsiemi di VxV"
  ```

  la relazione di corrispondenza tra grafo e sua rappresentazione e' data da: (u,v) e' un arco sse (u,v) e' in B.

Queste due rappresentazioni astratte possono essere specializzate scegliendo una delle rappresentazioni dei sottoinsiemi (bit-vector, sequenza senza ripetizioni, sequenza ordinata senza ripetizioni).

- se scegliamo di rappresentare i sottoinsiemi con dei bit-vetcors otteniamo la rappresentazione a matrice di adiacenza

  ```
  M: array [V,V] of boolean
  ```

  M[u,v] true sse (u,v) e' un arco.
- se scegliamo di rappresentare i sottoinsiemi con sequenze senza ripetizioni otteniamo la rappresentazione a sequenze di adiacenza (che a loro volta possono essere rappresentate con array, files, o liste linkate). Usando le sequenze senza riperizioni si rappresentano di fatto grafi ordinati.

Rappresentazione di alberi radicati

Oltre a rappresentare alberi radicati come grafi orientati, si puo' adottare una rappresentazione piu' compatta che sfruttare la proprieta' che ogni nodo, ad eccezione della radice r, ha un padre

```
P : array [V] of V
```

la corrispondenza tra albero e rappresentazione e' data da: u padre di v sse u=P[v] e v diverso da r. Non tutti gli array P corrispondono ad un albero, p.e. potrebbe rappresentare un grafo costituito da un albero ed un ciclo. Se n=|V|, si puo' convenire di enumerare i nodi da 1 ad n in modo che se v_i e' padre di v_j allora i < j (e quindi r=v_1).

Algoritmi su grafi

Descriveremo gli algoritmi usando la rappresentazione astratta dei grafi come array di insiemi di adiacenza.

- calcolo dello in-degree (grado entrante)

```
input: A: array [V] of "sottoinsiemi di V"
output IN: array [V] of integer

for v in V do {inizializzazione di IN}
  "azzera IN[v]"
for u in V do
  for v in A[u] do
    "incrementa IN[v]"
```

lo out-degree di u e' semplicemente la cardinalita' di A[u], se i sottoinsiemi sono rappresentati con un array allora la cardinalita' la si calcola in O(1).

- topological sorting di un grafo aciclico

```
input: A: array [V] of "sottoinsiemi di V" {grafo aciclico}
output O: array [V] of 0..|V| {t.c. 0 < O[u] distinti e O[u] < O[v] se (u,v) arc

IN <- in-degree     {vettore con in-degrees}
X0 <- {u|IN[u]=0}   {insieme di vertici con in-degree 0}
k  <- 0             {contatore del numero di ordine}

while "X0 non vuoto" do
  "rimuovi un elemento u da X0"
  "incrementa k"
  O[u] <- k
  for v in A[u] do  {rimuovi u dal grafo, aggiornando IN e X0}
    "decrementa IN[v]"
    if IN[v]=0 then "aggiungi v ad X0"
```

ESERCIZI: modificare l'algoritmo per effettuare anche il test di aciclicita'. Modificare l'algoritmo per calcolare la profondita' di ciascun nodo (=massimo tra le lunghezze dei cammini che arrivano a quel nodo).