Fondamenti di Informatica, A.A. 2012-2013

15/07/2013

Prova Pratica

Un grafo è definito da una coppia di insiemi `V` ed `E` in cui `V` è l'insieme dei nodi che compongono il grafo ed `E` è l'insieme degli archi che connettono due punti appartenenti a `V`. Un arco è rappresentato da una coppia di nodi.

Una esempio di grafo può essere il seguente:

$$\begin{aligned} V &= \{A, B, C, D\} \\ E &= \{(A, B), (A, C), (B, C), (B, D), (A, D)\} \end{aligned}$$

Un modo per rappresentare un grafo è attraverso una matrice di incidenza. Tale matrice è quadrata ed ogni sua riga/colonna rappresenta un nodo in `V`. La posizione `(i,j)` di questa matrice sarà 1 se esiste un arco che collega il nodo i con il nodo j, 0 in caso contrario.

Si realizzi una funzione chiamata `adjmat` che prende come parametri d'ingresso un insieme di nodi `V` ed un insieme di archi `E` e costruisce una matrice di incidenza che rappresenta il grafo $G = \{V, E\}$, definendo in maniera opportuna i parametri di ingresso e di uscita.

Bonus: Si realizzi una rappresentazione della matrice di incidenza più efficiente in termini di occupazione di memoria.

Svolgimento

L'elemento fondamentale del compito è la rappresentazione degli insiemi `V` e `E`; la soluzione dell'esercizio proposto può essere affrontata anche facendo delle ipotesi semplificative su tale rappresentazione. Dato che ai fini di questo esercizio i simboli A, B, C, D sono delle pure etichette senza ulteriore significato si può ad esempio considerare l'insieme `V` completamente descritto dalla sua dimensione n, avendo quindi sostituito ai nomi dei vari elementi il loro indice di posizione. In altre parole, consideriamo che sia

$$V = \{1, 2, \ldots, nv\}.$$

L'insieme `E` a questo punto è rappresentabile con una matrice a due colonne, in cui ciascuna riga contiene il primo ed il secondo nodo dell'arco; diventa quindi sufficiente scorrere tutti gli archi ed in corrispondenza di ciascuno forzare l'elemento della matrice di adiacenza, precedentemente inizializzata con degli zeri. Con queste ipotesi si ottiene il seguente codice:

```
function mat=adjmat(nv,edges)
  % Compute the adjacency matrix of a graph
  % represented with a set of vertices and edges.

  % First step: size of things.
  [nedges, nc]=size(edges);
  if (nedges>0)&&(nc~=2)
     error('Wrong edge size!');
  end
  mat=zeros(nv,nv);
```

```
  for k=1:nedges
      r=edges(k,1);
      c=edges(k,2);
      % The indices should be within the bounds.
      if ((1<=r)&&(r<=nv)&&(1<=c)&&(c<=nv))
          mat(r,c)=1;
      else
        printf("Error at row: %d: bad edge (%d,%d)\n",k,r,c)
      end
  end

end
```

Una soluzione più sofisticata è necessaria quando invece non si facciano ipotesi restrittive sulla struttura dell'insieme `V`, e quindi ci si trovi sotto condizioni piu' generali; ad esempio la nuova soluzione deve funzionare nel caso in cui

$$V = \{-1, 2, 5, 7, 9\}.$$

Il codice proposto di seguito ha quindi la necessità di confrontare gli elementi degli archi con l'elenco dei nodi per ricavarne la posizione, e lo fa usando gli operatori su array. Una ulteriore accortezza sta nell'ordinamento con eliminazione dei duplicati nell'insieme `V` in modo da essere sicuri che gli estremi degli archi diano ciascuno un confronto di uguaglianza con al più un elemento dell'insieme dei nodi.

```
function mat=adjmat(vertices,edges)
  % Compute the adjacency matrix of a graph
  % represented with a set of vertices and edges.

  % First step: size of things. Sort vertices and
  % eliminate duplicates (unlikely, but possible)
  vertices=unique(sort(vertices));
  nverts=length(vertices);
  [nedges, nc]=size(edges);
  if (nedges>0)&&(nc~=2)
     error("Wrong edge size!");
  end

  mat=zeros(nverts,nverts);

  for k=1:nedges
  % Since vertices have no duplicates the equality
  % is at most satisfied for one of them.
      r=(vertices==edges(k,1));
      c=(vertices==edges(k,2));
      if (any(r)&&any(c))
      % Here we have that the edge really mentions things in the
      % vertex set.
```

```
            mat(r,c)=1;
        else
          printf("Error at row: %d: bad edge \n",k);
          disp(edges(k,1:2));
        end
    end

end
```

Per quanto riguarda il bonus, il problema della rappresentazione con questa matrice riguarda la occupazione di memoria: se consideriamo un grafo per n molto grande, accade spesso che il numero effettivo di archi sia molto inferiore al massimo possibile n^2. Pertanto una soluzione opportuna cercherà di evitare la memorizzazione di grandi quantità di zeri.

Una rappresentazione possibile è quella in cui si usa una matrice di interi rettangolare in cui vengono memorizzati per ogni riga gli indici delle colonne cui corrispondono dei valori non-zero (o zero altrimenti). Considerando l'esempio proposto

$$G = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

questo si trasformerebbe in

$$G = \begin{pmatrix} 2 & 3 \\ 0 & 0 \\ 4 & 0 \\ 0 & 0 \end{pmatrix}.$$

In questo caso si passa da una matrice $n \times n$ ad una matrice $n \times k$ dove k è il massimo numero di elementi non-zero per riga, ovvero il massimo tra i gradi (numero di archi incidenti) per ciascun vertice. Molte altre rappresentazioni sono possibili; i concetti appena esposti divengono essenziali nel trattamento delle matrici sparse, ossia matrici che contengono molti coefficienti pari a zero, che compaiono ad esempio nella soluzione numerica dei problemi modellati con equazioni differenziali alle derivate parziali.