Problema grafo bipartito

Nico Fiorini

1 Problema 5.2, grafo bipartito

Risolvere il seguente problema:
Un grafo non orientato G=(V,E) si dice bipartito se è possibile partizionare i suoi vertici in due sottoinsiemi V_1 e V_2 tali che non esistono archi tra vertici nello stesso sottoinsieme: formalmente, se (u,v) $\in$ e, allora u $\in v_1$ e $u \in v_2$ o viceversa.

1. Dimostrare che un grafo è bipartito se e solo se non contiene cicli di lunghezza dispari, dove la lunghezza è il numero di archi nel ciclo.
2. Descrivere un algoritmo per verificare se un grafo connesso è bipartito. l'algoritmo deve esibire la partizione V_1 e V_2 nel caso di risposta affermativa, e un ciclo di lunghezza dispari nel caso di risposta negativa.
3. Discutere il tempo di esecuzione dell'algoritmo proposto.

Soluzione 1.

1.1 Lemma 1

Sia G=(V,E) un grafo non orientato, ogni componente connessa di G è bipartita, allora G è bipartito.

Dimostrazione

Supponiamo per assurdo che G non sia Bipartito, allora esiste un arco (x, y) tale che $x, y \in V_1$ o V_2. Ciò è assurdo poichè l'arco deve appartenere ad una componente connessa siccome y è raggiungibile da x, ciò è assurdo in quanto nega l'ipotesi di partenza ■

Teorema 1

Un grafo è bipartito se e solo se non contiene cicli di lunghezza dispari.

Dimostrazione

Procedo dimostrando un'implicazione per volta.

1° implicazione $\Rightarrow$

Si vuole dimostrare che se un grafo è bipartito $\Rightarrow$ non contiene cicli di lunghezza dispari.
Supponiamo per assurdo che esiste un ciclo di lunghezza dispari:
$\exists\, C = (w_1, w_2, ...w_n, w_1)$ tale che n = 2k+1 dove $k \in \mathbb{Z}$ e $w_i \in V \quad \forall\ 1 \leq i \leq n$

Senza perdita di generalità sia:

$$w_1 \in V_1$$
$$w_2 \in V_2$$

Sia $w_i \in v_1\ \ \forall i = 2k+1 \rightarrow w_n \in V_1$
È assurdo in quanto $(w_n, w_1) \in E$ e $w_n, w_1 \in V_1$

2° implicazione $\Leftarrow$

È sufficiente mostrare quest'implicazione per un grafo connesso, se il grafo non è connesso, basta ripetere la dimostrazione per ogni componente connessa di G in virtù del lemma 1.

Sia $G = (V, E)$ un grafo senza cicli di lunghezza dispari
Sia $w \in G$, posso partizionare il grafo in due sottoinsiemi:

$$V_1 = \{\ v \in V\ |d(v, w)\ \text{ è pari }\}$$
$$V_2 = \{\ v \in V\ |d(v, w)\ \text{ è dispari }\}$$
$$V_1 \cap V_2 = \emptyset \quad V_1 \cup V_2 = V$$

Voglio dimostrare che questa partizione è di un grafo bipartito.
Supponiamo per assurdo che questa partizione non lo sia, quindi deve esistere un arco che collega 2 vertici dello stesso insieme, formalmente :

$$\exists(a,b) \in E \text{ tale che } a,b \in V_1 \text{ oppure } a,b \in V_2$$

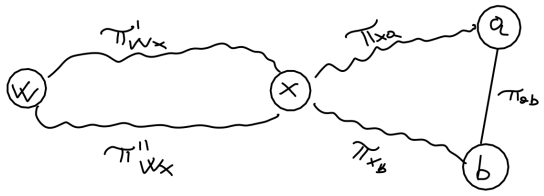
Senza perdita di generalità, sia $a = w \Rightarrow d(a,w) = 0$ quindi $a \in V_1$ ed anche $b \in V_1 \rightarrow d(w,b) = d(a,b) = 2k$.
È assurdo poichè:

$$1 = d(a,b) = 2k$$

Quindi si conclude che $a \neq b \neq w$.

Sia ora π_{wa} e π_{wb} i due cammini minimi che vanno rispettivamente da w ad a, e da w a b. I due cammini hanno almeno un vertice in comune, in quanto w è comune a tutti e due i cammini, tuttavia può avere anche altri vertici in comune.

Sia x l'ultimo vertice in comune nei due cammini, una rappresentazione grafica è riportata nell'immagine seguente:



La lunghezza dei due cammini π_{wa} e π_{wb} sono tutti e due pari o tutti e due dispari, poichè a e b appartengono allo stesso insieme, e siccome sono cammini minimi, la lunghezza dei cammini corrisponde alla distanza tra gli estremi dei cammini. Grazie a questa proprietà possiamo dire che la lunghezze di $\pi'_{w,x}$ e $\pi''_{w,x}$ sono uguali, quindi le lugghezze di π_{xa} e π_{xb} continuano ad essere tutte e due pari o tutte e due dispari.

Calcoliamo la lunghezza del seguente ciclo:

$$C = (\pi_{ax}, \pi_{xb}, \pi_{b,a})$$
$$l(C) = l(\pi_{ax}) + l(\pi_{xb}) + l(\pi_{b,a})) = d(a,x) + d(x,b) + d(b,a)$$

Sappiamo che $l(\pi_{ax}), l(\pi_{xb})$ sono tutte e due pari o tutte e due dispari, di conseguenza $d(a,x) + d(x,b) = 2k$ inoltre $d(b,a) = 1$ siccome sono vertici connessi da un arco per ipotesi, quindi:

$$l(C) = 2k + 1$$

È assurdo poichè in partenza abbiamo supposto che non esistono cicli di lunghezza dispari, quindi possiamo concludere che il partizionamento non può avere archi che collegano due vertici dello stesso insieme dal momento in cui ipotizziamo che non abbiamo cicli di lunghezza dispari, quindi il partizionamento trovato è di un grafo bipartito. ■

Ricapitolando, abbiamo supposto che non esistono cicli di lunghezza dispari, abbiamo partizionato il grafo in due sottoinsiemi, facendo questo abbiamo provato a trovare un arco che unisce due vertici dello stesso insieme, ma facendo questo arriviamo all'assurdo di non soddisfare l'ipotesi di partenza, quindi abbiamo trovato che il grafo è bipartito se non ha cicli di lunghezza dispari.

Soluzione 2.

Basandoci sul teorema precedente, è possibile realizzare un algoritmo che dato un grafo in input, è in grado di riconoscere se è bipartito provando a trovare un ciclo di lunghezza dispari nel grafo.

Possiamo ricavare l'algoritmo basandoci su una visita BFS, in quanto l'albero della visita BFS gode di alcune proprietà utili per il nostro problema.

Lemma 2

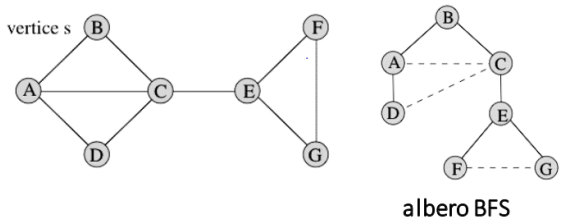
Siano dati un grafo non orientato e connesso $G = (V, E)$, un vertice s in G, ed un albero T prodotto dall'algoritmo visitaBFS(s). Per ogni vertice $v \in T$, risulta $l(v) = d_{sv}$

Questa lemma è particolarmente utile per sfruttare il partizionamento usato nella dimostrazione del teorema 1, in quanto possiamo marcare i vertici con livello di profondità pari dell'alberto BFS nell'insieme V_1, e marchiamo nell'insieme V_2 i vertici con profondità dispari nell'albero BFS.

Proprietà 1

Ogni arco di un grafo non orientato G su cui si effettua la visita in ampiezza può essere classificato in tre gruppi rispetto all'albero BFS prodotto:

1. archi dell'albero BFS.
2. archi tra vertici allo stesso livello dell'albero BFS.
3. archi tra livelli adiacenti dell'albero BFS.



L'arco di tipo 2, è un arco che unisce due vertici della stessa profondità dell'albero BFS,quindi unisce due vertici dello stesso insieme, di conseguenza basta controllare con una variante dell'algortimo visitaBFS, che non ci siano archi di tipo 2, nel momento in cui abbiamo un arco di tipo 2, siamo certi della presenza di un ciclo negativo, altrimenti se si arriva a visitare tutto il grafo senza incontrare archi di tipo 2, allora l'algoritmo non ha archi che uniscono due vertici dello stesso insieme.

```
1:  bool isBipartite(Grafo G,Node s)
2:  Marca tutti i vertici di G come inesplorati
3:  Marca s come esplorato
4:  Assegna ad s il livello 0
5:  T ← albero formato dalla radice s
6:  Coda F
7:  F.enqueque(s)
8:  while not F.isEmpty() do
9:     u ← F.dequeque()
10:    l = livello di u
11:    visitau
12:    for all arco(u, v) ∈ G do
13:       if v è inesplorato then
14:          marca v come esplorato
15:          F.enqueque(v)
16:          rendi u padre di v in T
17:          assegna a v il livello l + 1
18:       end if
19:       if l = livello di v then
20:          StampaCicloDispari(u, v)
21:          return false
22:       end if
23:    end for
24: end while
25: StampaInsiemi(s)
26: return true
```

Dati i due vertici uniti da un arco di tipo 2 della proprietà 1, è possibile stampare i valori dei nodi contenuti nel ciclo, risalendo di padre in padre nei 2 vertici, fino a quando non si incontra il vertice in comune.

```
1: void StampaCicloDispari(Node u,Node v)

2: bool common := false
3: Sia X un array ausiliario
4: int i:=0
5: while not common do
6:    print  valore di u
7:    X[i] = valore di v
8:    i++
9:    u ← parent(u)
10:   v ← parent(v)
11:   if valore di u = valore di v then
12:      print  valore di u
13:      common:=true
14:   end if
15: end while
16: while I do
17:   print  valore di v
18:   i−−
19: end while
```

Nel caso il grafo è bipartito, cioè non si hanno archi di tipo 2, si stampano i due sottoinsiemi tramite la funzione StampaInsiemi(Node s). La funzione parte dal vertice s, ed esegue una visita dei nodi nell'albero BFS, stampando i nodi.

```
1: void StampaInsiemi(Node s)

2: Sia X un array ausiliario
3: int i:=0
4: Coda C
5: C.enqueque(s)
6: print  Nodi dell'insieme V1 :
7: while not C.isEmpty() do
8:    u ← C.dequeque
9:    if u ≠ null then
10:      if valore u % 2 ==0 then
11:         print  valore di u
12:      else
13:         X[i++] = valore di u
14:      end if
15:   end if
16:   for all i ∈ figli di u do
17:      C.enqueque(i)
18:   end for
19: end while
20: int j=0;
21: print  Nodi dell'insieme V2
22: while j < i do
23:   print  X[j++]
24: end while
```

2 Soluzione 3.

Il costo computazionale dell'algoritmo "isBipartite" trovato per la Soluzione 2 è lo stesso ottenuto per una visita di un grafo, $O(m + n)$ nel caso i grafo sono implementati con liste di adiacenza o liste di incidenza. La stampa del ciclo dispari o degli insiemi non incide sull'andamento asintotico, poichè sono funzioni che vengono eseguite solo una volta, alla fine dell'algoritmo e tutte e due hanno costo computazionale $O(n)$ nel caso peggiore.