Algoritmi e Strutture Dati (Complementi)

Appello straordinario del 23/06/2003 – Prof. Giancarlo Mauri

Soluzioni

Esercizio 1. Data una sequenza finita $a_1, a_2, \ldots, a_n$ di interi positivi, si vuole trovare la lunghezza della più lunga sottosequenza crescente formata da numeri dispari. Scrivere un algoritmo di programmazione dinamica (e la corrispondente equazione di ricorrenza) che consenta di risolvere il problema.

Soluzione. Detta $A = a_1, a_2, \ldots, a_n$ la sequenza di numeri data in ingresso, per ogni $i \in \{1, 2, \ldots, n\}$ indichiamo con A_i la sottosequenza $a_1, \ldots, a_i$.

Definiamo un vettore $L[1..n]$ di interi, dove per $i \in \{1, 2, \ldots, n\}$ il valore $L[i]$ è la lunghezza della più lunga sottosequenza crescente della sottosequenza A_i, formata da numeri dispari, che termina con a_i. È facile verificare che vale:

$$L[1] = \begin{cases} 1 & \text{se } a_1 \text{ è dispari} \\ 0 & \text{se } a_1 \text{ è pari} \end{cases}$$

mentre, per $i > 1$:

$$L[i] = \begin{cases} 0 & \text{se } a_i \text{ è pari} \\ 1 + \max\{L[k] \mid 1 \leq k < i,\ a_k < a_i\} & \text{se } a_i \text{ è dispari} \end{cases}$$

Poiché può accadere che l'insieme $\{L[k] \mid 1 \leq k < i,\ a_k < a_i\}$ sia vuoto, poniamo per definizione $\max \emptyset = 0$, altrimenti in questo caso il valore di $L[i]$ risulterebbe indefinito. Si noti inoltre che nella seconda condizione dell'equazione che definisce $L[i]$ non è necessario controllare che anche a_k sia dispari. Infatti, se a_k è pari vale $L[k] = 0$; anche se tale valore fosse il massimo, avremmo $L[i] = 1 + 0 = 1$, che è corretto.

Una volta calcolati i valori $L[1], L[2], \ldots, L[n]$, la soluzione al problema proposto è data da:

$$\max_{1 \leq i \leq n} L[i]$$

L'algoritmo che calcola i valori $L[1], L[2], \ldots, L[n]$ è il seguente:

```
LONGEST-ODD-SUBSEQUENCE
if a_1 mod 2 = 0
   then L[1] ← 0
   else L[1] ← 1
for i ← 2 to n
```

```
do if a_i mod 2 = 0
      then L[i] ← 0
      else max ← 0
           for k ← 1 to i − 1
              do if (a_k < a_i) and (L[k] > max)
                    then max ← L[k]
           L[i] ← 1 + max
```

La complessità in tempo dell'algoritmo proposto è $O(n^2)$.

Esercizio 2. Sia $S = \{1, \ldots, 1000\}$ l'insieme dei primi 1000 interi positivi, e sia I la famiglia di sottoinsiemi di S tale che $A \subseteq S$ appartiene a I se e solo se:

$$\forall\, a \in A \qquad a \equiv 0 \pmod 3$$

ovvero se e solo se ogni elemento di A è divisibile per 3.

Dimostrare che la coppia (S, I) è un matroide oppure, nel caso in cui non lo sia, fornire un controesempio.

Soluzione. La coppia (S, I) proposta è un matroide. Infatti, valgono le seguenti proprietà.

- S è un insieme finito e non vuoto.
- I è un insieme finito (contiene al più tutti i sottoinsiemi di S) e non vuoto (ad esempio, $\emptyset \in I$, $\{3\} \in I$, ...).
- Vale la **proprietà della ereditarietà**: dati $A \in I$ e $B \subseteq A$, vale $B \in I$.

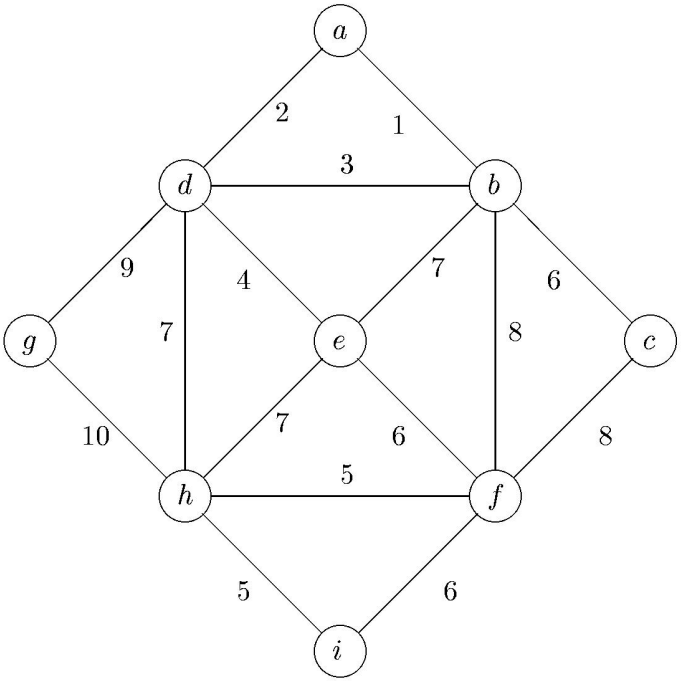
  Infatti $A \in I$ significa che ogni elemento di A è divisibile per 3. Tutti gli elementi di un qualunque sottoinsieme B di A sono chiaramente divisibili per 3, e quindi $B \in I$.
- Vale la **proprietà dello scambio**: dati $A, B \in I$ tali che $|A| < |B|$, $\exists\, x \in B \setminus A$ tale che $A \cup \{x\} \in I$.

  $A, B \in I$ significa che tutti gli elementi di A e di B sono divisibili per 3. Poiché B contiene più elementi di A, esiste sicuramente un numero x che appartiene a B ma non appartiene ad A. Dato che $x \in B$, x è divisibile per 3; pertanto tutti gli elementi di $A \cup \{x\}$ sono divisibili per 3, e quindi $A \cup \{x\} \in I$.

Osserviamo che (S, I) ammette un unico elemento linearmente indipendente massimale, che è:

$$\{3k \mid k \in \{1, 2, \ldots, 333\}\}$$

Esercizio 3. Applicare l'algoritmo di Kruskal al grafo pesato mostrato in figura. Mostrare le operazioni eseguite sulle varie strutture dati usate dall'algoritmo.



Soluzione. Ricordiamo che l'algoritmo di Kruskal crea un albero di copertura minimo di un grafo pesato. Le strutture dati utilizzate dall'algoritmo sono:

- una lista, che chiamiamo A, di archi che appartengono all'albero di copertura minimo;
- una collezione di insiemi disgiunti, sui quali si opera tramite le operazioni Make–Set, Find–Set e Union tipiche delle strutture union–find.

Inizialmente abbiamo $A = \emptyset$ e $|V|$ insiemi singoletto, uno per ogni vertice del grafo.

Quindi, si ordinano gli archi del grafo in ordine non decrescente rispetto al peso, ottenendo ad esempio:

$$(a,b),(a,d),(b,d),(d,e),(f,h),(h,i),(b,c),(f,i),$$
$$(e,f),(b,e),(d,h),(e,h),(b,f),(c,f),(d,g),(g,h)$$

A questo punto si considerano gli archi uno alla volta, nell'ordine appena individuato; se i vertici dell'arco appartengono a due insiemi disgiunti diversi, si aggiunge l'arco ad A e si uniscono i due insiemi disgiunti. Pertanto abbiamo:

Arco considerato	Insiemi disgiunti	A
–	$\{a\},\{b\},\{c\},\{d\},\{e\},\{f\},\{g\},\{h\},\{i\}$	$\emptyset$
(a,b)	$\{a,b\},\{c\},\{d\},\{e\},\{f\},\{g\},\{h\},\{i\}$	(a,b)
(a,d)	$\{a,b,d\},\{c\},\{e\},\{f\},\{g\},\{h\},\{i\}$	$(a,b),(a,d)$
(b,d)	(uguale)	(uguale)
(d,e)	$\{a,b,d,e\},\{c\},\{f\},\{g\},\{h\},\{i\}$	$(a,b),(a,d),(d,e)$
(f,h)	$\{a,b,d,e\},\{c\},\{f,h\},\{g\},\{i\}$	$(a,b),(a,d),(d,e),(f,h)$
(h,i)	$\{a,b,d,e\},\{c\},\{f,h,i\},\{g\}$	$(a,b),(a,d),(d,e),(f,h),$ (h,i)
(b,c)	$\{a,b,c,d,e\},\{f,h,i\},\{g\}$	$(a,b),(a,d),(d,e),(f,h),$ $(h,i),(b,c)$
(f,i)	(uguale)	(uguale)
(e,f)	$\{a,b,c,d,e,f,h,i\},\{g\}$	$(a,b),(a,d),(d,e),(f,h),$ $(h,i),(b,c),(e,f)$
(b,e)	(uguale)	(uguale)
(d,h)	(uguale)	(uguale)
(e,h)	(uguale)	(uguale)
(b,f)	(uguale)	(uguale)
(c,f)	(uguale)	(uguale)
(d,g)	$\{a,b,c,d,e,f,g,h,i\}$	$(a,b),(a,d),(d,e),(f,h),$ $(h,i),(b,c),(e,f),(d,g)$
(g,h)	(uguale)	(uguale)