Lab. Programmazione (CdL Informatica) & Informatica (CdL Matematica) a.a. 2022-23

Monica Nesi

Università degli Studi dell'Aquila

4 Ottobre 2022

Algoritmi e diagrammi di flusso

Per i problemi che saranno trattati nelle pagine seguenti fare riferimento alla dispensa "Esercizi su Algoritmi e Diagrammi di Flusso" e le relative soluzioni con i diagrammi di flusso in http://people.disim.univaq.it/monica/Lab-Progr/lezioni.html.

In questa lezione verrà considerato il problema di calcolare il *massimo* (risp. *minimo*) di un insieme di elementi, su cui è definita una *relazione di ordinamento*.

Massimo di due numeri

Esercizio 1. Dati due numeri naturali m ed n, calcolare il massimo (risp. minimo) tra m ed n.

Input: $m, n \in \mathbb{N}$
Output: $max(m, n)$

Operazioni elementari della macchina:

- operazioni di input/output per acquisizione dati e restituzione risultato (e.g. lettura in ingresso e scrittura/stampa in uscita);
- operazioni di confronto tra numeri ($>$, $\geq$, $<$, $\leq$, $=$, etc.).

Vedere il diagramma di flusso.

Massimo di due numeri (cont.)

Osservazioni

1. Nella soluzione data viene utilizzato l'operatore $>$, ma è possibile usare anche uno qualsiasi degli altri operatori di confronto (con opportune modifiche all'algoritmo).

2. Nel caso in cui $m = n$, si restituisce uno qualsiasi tra m ed n come valore massimo.

3. L'algoritmo per calcolare il minimo tra m ed n può essere dato in modo simile.

4. L'algoritmo si estende facilmente al massimo/minimo di numeri interi, decimali, etc., assumendo di avere le relative operazioni di confronto.

Massimo di tre numeri

Esercizio 2. Dati tre numeri naturali m, n e p, calcolare il massimo (risp. minimo) dei tre numeri.

Input: $m, n, p \in \mathbb{N}$
Output: $max(m, n, p)$

Le operazioni elementari necessarie sono le stesse dell'esercizio precedente.

L'algoritmo proposto fa confronti successivi tra i numeri a due a due (vedere diagramma).

In modo simile può essere dato l'algoritmo per calcolare il minimo di tre numeri.

Massimo di una sequenza di numeri

Esercizio 3. Data una sequenza finita (non vuota) di numeri interi (terminata da uno 0)

$$x_1 \; x_2 \; \ldots \; x_n \; 0$$

calcolare il massimo (risp. minimo) della sequenza.

N.B. Qui supponiamo che ogni *sequenza* di elementi sia

- finita (i.e. costituita da un numero finito di elementi),
- non vuota (i.e. esiste almeno un elemento nella sequenza diverso da 0 prima di uno 0 di fine sequenza)
- terminata da uno 0 (i.e. il numero 0 viene usato come marcatore di fine sequenza ed eventuali numeri dopo uno 0 sono ignorati).[1]

[1] Nei linguaggi di programmazione esistono costrutti per verificare se si è alla fine di una sequenza, e lo 0 tornerà ad essere un numero come gli altri.

Massimo di una sequenza di numeri (cont.)

Input: $x_1\ x_2\ \ldots\ x_n\ 0$ con $x_i \in \mathbb{Z}$ ed $x_i \neq 0$ per ogni $i = 1, \ldots, n$
Output: $max(x_1, x_2, \ldots, x_n)$

Possibile algoritmo: esaminare *tutti* i numeri della sequenza e confrontarli con un valore che rappresenta il *massimo corrente*, ovvero il massimo della porzione di sequenza già esaminata.

Hp. utilizziamo due *locazioni (o celle) di memoria*, dove leggere e scrivere valori interi, riferite simbolicamente con i nomi `max` ed `n`.

La locazione `max` funge da *accumulatore*, i.e. in tale cella viene memorizzato il valore del massimo corrente.

Nella locazione `n` viene copiato l'elemento della sequenza considerato ad un certo passo dell'algoritmo per confrontarlo con (il valore di) `max`.

Massimo di una sequenza di numeri (cont.)

Inizializzazione di `max` ed `n`.

La sequenza in ingresso è non vuota ed il suo primo elemento (che esiste ed è diverso da 0) è usato per inizializzare il contenuto della locazione `max`.

N.B. L'inizializzazione di `max` con il valore 0 può non essere corretta su $\mathbb{Z}$, e.g. nel caso in cui i numeri della sequenza siano tutti negativi.

La locazione `n` può essere inizializzata con il primo elemento della sequenza o con l'elemento successivo.

Massimo di una sequenza di numeri (cont.)

Idea dell'algoritmo

Ripetere i seguenti passi fino a quando non si raggiunge la fine della sequenza:
- si seleziona un elemento x_i della sequenza (in genere si parte da x_1 fino a quando si raggiunge il primo 0) copiandolo nella locazione `n`;
- si confronta il valore di `n` con quello di `max`;
- se `n` risulta (strettamente) maggiore di `max`, il valore di `max` viene aggiornato con il valore di `n`, altrimenti non viene apportata alcuna modifica su `max`.

In modo simile può essere dato l'algoritmo per calcolare il *minimo* di una sequenza di numeri.

Massimo di una sequenza di numeri (cont.)

Operazioni elementari della macchina:

- operazioni di input/output;
- lettura/scrittura in memoria;
- selezione del primo elemento di una sequenza e dell'elemento successivo;
- operazioni di confronto su $\mathbb{Z}$.

Nel caso in cui il massimo (risp. minimo) compaia più di una volta nella sequenza, basta restituire una delle sue occorrenze.

Esempio: eseguire l'algoritmo descritto dal diagramma di flusso nel caso della sequenza `4 -1 5 7 2 7 -3 0`.

Sequenze di caratteri: stringhe

Una *stringa* è definita come una sequenza di caratteri.

Esempi di stringhe: `bbccc`, `a1` ed `fg47`.

La *lunghezza* di una stringa s è data dal numero di caratteri in s.

La *stringa vuota* (ovvero la stringa con zero caratteri) è denotata con il simbolo ϵ.

Le stringhe possono essere *concatenate* per formare stringhe più lunghe semplicemente *giustapponendo* le stringhe una dopo l'altra. Esempio: la concatenazione delle stringhe `a1` ed `fg47` è la stringa `a1fg47`.

La concatenazione è associativa, ma non è commutativa.

La stringa vuota ϵ è l'*elemento neutro o identità* per la concatenazione, ovvero per ogni stringa s si ha: $s\,\epsilon = \epsilon\,s = s$.

Stringa di lunghezza massima in una sequenza

Esercizio 4. Data una sequenza finita (non vuota) di stringhe (terminata dalla stringa vuota ϵ), calcolare la stringa di lunghezza massima (risp. minima) della sequenza.

Input: $s_1\ s_2\ \ldots\ s_k\ \epsilon$
Output: la stringa s_j di lunghezza massima ($j \in \{1, \ldots, k\}$)

Problema simile a quello dell'esercizio precedente.

L'unica differenza è il tipo di elementi considerati.

Ciò porta a richiedere operazioni elementari diverse, in quanto le stringhe sono oggetti in genere più complessi dei numeri interi, ma l'algoritmo non cambia.

Stringa di lunghezza massima in una sequenza (cont.)

Importante: fare *astrazione* sul tipo degli elementi:

il diagramma di flusso/algoritmo per questo problema ha la stessa *struttura* di quello dell'esercizio precedente.

Hp. due locazioni di memoria atte a contenere valori di tipo stringa, riferite simbolicamente con `max` (contenente la stringa massima corrente) ed `s` (contenente la stringa s_i in esame).

Modifiche nel diagramma di flusso:

- rimpiazzare numeri con stringhe ed `n` con `s`;
- $s \neq \epsilon$ per il test di fine sequenza;
- `lung(s) > lung(max)` per il confronto tra le lunghezze, dove `lung` denota la funzione che calcola la lunghezza di una stringa.

Stringa di lunghezza massima in una sequenza (cont.)

Esempio: eseguire l'algoritmo descritto dal diagramma di flusso nel caso della sequenza `abc dd alfa q gamma lpr` ϵ.

Cosa succede se vi sono più stringhe di lunghezza massima (risp. minima) nella sequenza in ingresso?
Esempio: `abc r alfa q beta lprm` ϵ.

In tal caso la specifica del problema può richiedere di restituire una stringa in particolare tra quelle di lunghezza massima (risp. minima), per esempio quella *più a destra* o quella *più a sinistra* nella sequenza data.

Come calcolare tali stringhe di lunghezza massima (risp. minima)?