UNIVERSITÀ DEGLI STUDI DI MILANO

DIPARTIMENTO DI INFORMATICA

Tutorato di programmazione

Piano Lauree Scientifiche - Progetto di Informatica

Goal che richiedono la combinazione di piani

In questa scheda si parla della combinazione di piani iterativi,
già presentati nella scheda "Goal che richiedono la composizione di piani".

Ultimo aggiornamento 27 marzo 2021 - 22:39:55

Autori:
Violetta Lonati (coordinatrice del gruppo), Anna Morpurgo e Umberto Costantini

Hanno contribuito alla revisione del materiale gli studenti tutor:
Alessandro Clerici Lorenzini, Alexandru David, Andrea Zubenko, Davide Cernuto, Francesco Bertolotti, Leonardo Albani, Luca Tansini, Margherita Pindaro, Umberto Costantini, Vasile Ciobanu, Vittorio Peccenati.

Si ringraziano per i numerosi spunti:
Carlo Bellettini, Paolo Boldi, Mattia Monga, Massimo Santini e Sebastiano Vigna.

1 Composizione di piani

Quasi sempre le specifiche di un programma richiedono lo svolgimento di diversi compiti o la soluzione di diversi sotto-problemi. A ciascuno di questi corrisponde un *goal* e quindi un piano nel programma. Ci sono diversi modi in cui i piani per i sotto-problemi vengono composti nella soluzione di un problema, che dipendono dal problema stesso.

- **Concatenazione** (in inglese *abutment*): i piani vengono applicati uno dopo l'altro; ciascun piano appare nel codice integralmente, come un'unità già completa.
- **Annidamento** (in inglese *nesting*): i piani vengono applicati uno dentro l'altro; un esempio tipico è nel doppio ciclo usato per scandire le righe di una matrice.
- **Fusione** (in inglese *merge* o *interleaving*): i due piani prevedono azioni comuni che possono essere svolte una volta sola e quindi i due blocchi di istruzioni corrispondenti vengono fusi in un unico blocco di istruzioni; spesso per fondere due piani è necessario scomporli e ricomporli in maniera opportuna, a seconda del problema da risolvere.

La fusione e l'annidamento di due piani sono i due tipi di composizione in cui il modo di comporre i piani non è sempre immediato, spesso occorre un po' di progettazione per comporre correttamente le sotto-soluzioni.

Illustriamo ora i diversi tipi di composizione attraverso esempi concreti in cui vengono applicati.

1.1 Concatenazione di piani

Specifiche. Data una lista di numeri, stampare la somma dei numeri della lista che superano la metà del massimo della lista stessa. Ad esempio se la lista contiene i numeri 4, 10, 2, 3, 5, 7, 6, il risultato sarà 23 (la somma di 10, 7 e 6).

Per risolvere il problema, dobbiamo prima calcolare il *massimo*, esaminando tutta la lista, e per questo *goal* possiamo sfruttare il *piano per la ricerca del valore estremo*. Un volta calcolato il massimo, dobbiamo calcolare la *somma* dei numeri che superano la metà del massimo, abbiamo cioè un *calcolo di un totale*, per il quale sarà necessario confrontare ogni elemento della lista con la metà del massimo, e quindi ripercorrere tutta la lista.

Il *piano per la ricerca del valore estremo* e quello per *calcolo di un totale* andranno quindi concatenati uno dopo l'altro. I due piani possono addirittura essere implementati in due funzioni separate, come nel codice qui sotto, che verranno invocate una dopo l'altra (`massimo` alla riga 4 e `sommaSeOltre` alla riga 5)

```
1  func main() {
2    lista := []int{4, 10, 2, 3, 5, 7, 6}
3    var max, somma int
4    max = massimo(lista)
5    somma = sommaSeOltre(lista, max/2)
6    fmt.Println(somma)
7  }
8
9  func massimo( lista []int ) int{
10   max := lista[0]
11
12   for _, el := range lista {
13    if el > max {
14     max = el
15    }
16   }
17   return max
18 }
19
```

```
20 func sommaSeOltre(lista []int, soglia int) int {
21   var somma = 0
22   for _, el := range lista {
23     if el > soglia {
24       somma += el
25     }
26   }
27   return somma
28 }
```

1.2 Annidamento di piani

Specifiche. Data una lista di stringhe, stabilire: i) quante parole contengono almeno una cifra e ii) il numero delle cifre contenute in ciascuna stringa.

Per risolvere il problema occorre usare il *piano per il conteggio* sia per il primo *goal* che per il secondo *goal*: per contare le stringhe che contengono cifre e, per ogni stringa, per contare le cifre all'interno della stringa stessa.

Nella porzione di codice seguente si vede che il *piano per il conteggio delle cifre* (righe 3-9) è annidato nel *piano per il conteggio delle stringhe che hanno cifre* (righe 1-2, 10-13): questo (esterno) itera sulle strighe della lista e quello (interno), per ciascuna stringa, sui suoi caratteri (rune). Il contatore `contaCifre`, che fa parte del piano *interno*, viene inizializzato a 0 ogni volta che si inizia a trattare una nuova stringa, quindi ad ogni iterazione del ciclo *esterno*.

```
1   contaStringhe := 0
2   for _, stringa := range os.Args[1:] {
3     contaCifre := 0
4     for _,lettera := range stringa {
5       if unicode.IsDigit(lettera) {
6         contaCifre++
7       }
8     }
9     fmt.Println(stringa, "contiene", contaCifre, "cifre.")
10    if contaCifre > 0 {
11      contaStringhe++
12    }
13  }
14  fmt.Println("Ho trovato", contaStringhe, "parole con almeno una cifra.")
```

1.3 Fusione di piani

Specifiche. Data una lista di voti, calcolare il voto massimo e la somma totale dei voti.

Quando sono richieste più elaborazioni su una stessa lista di dati, spesso non occorre ripetere la scansione della lista più volte, ma è possibile "fondere" i diversi piani in un unico ciclo.

In questo caso i piani richiesti sono la *ricerca del valore estremo* e il *calcolo di un totale*. La funzione `maxSum` sotto riportata risolve il problema: il piano per la *ricerca del valore estremo* (massimo) è riconoscibile alle righe 12-13-14-15-17, quello per il *calcolo di un totale* alle righe 12-16-17. Si osservi che le righe 12 e 17 sono comuni ad entrambi i piani: i piani sono stati fusi insieme in modo da scorrere una volta sola la lista.

```
1 func main() {
2   lista := []int{24, 29, 22, 23, 25, 27, 18}
3   var max, somma int
4   max, somma = maxSum(lista)
```

```
5     fmt.Println("massimo:", max, ", somma:", somma)
6   }
7
8   func maxSum( lista []int ) (int, int){
9     sum := 0
10    max := lista[0]
11
12    for _, el := range lista {
13        if el > max {
14          max = el
15        }
16        sum += el
17      }
18    return max, sum
19  }
```

1.4 Osservazione

Se i piani considerati sono piani iterativi, allora ognuno di essi avrà un ciclo. In questo caso sarà possibile riconoscere la modalità di composizione considerando quanti cicli sono presenti e, nel caso siano più di uno, come sono posti uno rispetto all'altro, ovvero:

- concatenazione: sono presenti due cicli, uno dopo l'altro;
- annidamento: sono presenti due cicli, uno dentro l'altro;
- fusione: è presente un unico ciclo.

2 Test di verifica della lettura di “Composizione di piani”

Prima di rispondere alle domande che seguono e svolgere gli esercizi della prossima scheda, leggete con attenzione la sezione precedente sulla omposizione di piani.

1. Quali sono i tipi di composizione possibili? Scegli tra le opzioni del seguente elenco le risposte corrette:
   a) composizione gerarchica
   b) annidamento
   c) fusione
   d) inquadramento
   e) ribaltamento
   f) concatenazione
   g) composizione funzionale
2. Nell’esempio 1.1, per calcolare la somma dei numeri che superano la metà del massimo quali piani sono necessari?
   - calcolo di un totale / ricerca lineare / ricerca del valore estremo / elaborazione su valori adiacenti
3. Nell’esempio 1.2 sull’annidamento di piani, che piani occorrono al fine di contare quante parole contengono almeno una cifra e determinare il numero di cifre in ciascuna parola?
   - calcolo di un totale / ricerca lineare / elaborazione su valori adiacenti / conteggio
4. Nell’esempio 1.3 sulla fusione di piani, quali sono i due piani implementati nella funzione `maxSum`?
   - calcolo di un totale / ricerca del valore estremo / elaborazione su valori adiacenti / conteggio
5. Trascrivi le istruzioni comuni ad entrambi i piani dell’esempio 1.3.
6. Se considero la concatenazione di due piani iterativi, quanti cicli avrò?
   a) due cicli, uno dentro l’altro
   b) un unico ciclo
   c) due cicli, uno dopo l’altro
7. Se considero l’annidamento di due piani iterativi, quanti cicli avrò?
   a) due cicli, uno dentro l’altro
   b) un unico ciclo
   c) due cicli, uno dopo l’altro
8. Se considero la fusione di due piani iterativi, quanti cicli avrò?
   a) due cicli, uno dentro l’altro
   b) un unico ciclo
   c) due cicli, uno dopo l’altro