Laboratorio di algoritmi e strutture dati

Scheda di esercizi su: ricorsione*

Docente: Violetta Lonati

1 Funzione ricorsiva semplice

Considerate la seguente funzione, che deve restituire il prodotto dei suoi due argomenti Cosa si deve scrivere, nel caso base, al posto di `AAAAA` e di `BBBBB`?

Listing 1:

```
func multiply(x, y int) int {
  if AAAAAA {
    return BBBBB
  } else {
    return x + multiply(x, y-1)
  }
}
```

2 Tracciatura di funzione ricorsiva

Considerate la seguente funzione, che deve calcolare il valore massimo contenuto nel vettore `numbers`. La funzione max calcola il massimo tra i suoi due argomenti.

Listing 2:

```
func largest(numbers []int) int {
  n := len(numbers)
  if -------- {
    return numbers[0]
  }
  return max( numbers[n-1], largest(  --------------  ) )
}
```

*Ultimo aggiornamento: 13 ottobre 2022 - 09:54:14

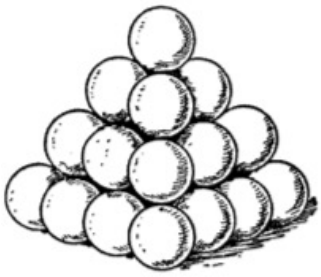
1. Come deve essere completato il caso base?

   Assumete d'ora in poi che il vettore `numbers` contenga i valori
   `[ 1, 2, 5, 7, -2, 10, 9, 21, 3, 8 ]`.

2. Durante l'esecuzione della chiamata `largest(numbers)`, considerate la chiamata ricorsiva che termina per prima. Qual è l'argomento passato in questa chiamata e quale valore restituisce la funzione?
3. Con quali argomenti viene eseguita per la prima volta la funzione `max`
4. E con quali argomenti viene eseguita l'ultima volta la funzione `max`?

3 Sassi sferici

Dei sassi sferici sono ammucchiati a formare una piramide, con un sasso in cima, posto al centro di un quadrato formato da 4 sassi (2 per lato), posti a loro volta sopra un quadrato formato da 9 sassi (3 per lato), e così via.



Scrivete una funzione ricorsiva `sassi(height int) int` che, data l'altezza height della piramide, restituisca il numero di sassi che la compongono.

Ad esempio: `f(1)` deve restituire 1 e `f(2)` deve restituire 5.

4 Ricorsione e iterazione

Considerate questa funzione ricorsiva `f_rec`

Listing 3:

```
1 func f_rec(n int) uint64 {
2   if n == 1 || n == 2 {
3     return 1
4   }
5   return f_rec(n-1) + f_rec(n-2)
6 }
```

Senza eseguire la funzione al computer, rispondete alle seguenti domande:

1. Cosa restituisce la funzione `f_rec(7)`?
2. Perché `n` è dichiarato come intero mentre il valore restituito è di tipo `uint64`?
3. Riassumete a parole cosa restituisce la funzione se riceve come argomento un intero positivo *n* maggiore di 0.

Considerate ora le due funzioni `f_iter1` e `f_iter2`

Listing 4:

```
1  func f_iter1(n int) uint64 {
2    var f uint64
3    var f1, f2 uint64 = 1, 1
4
5    if n == 2 || n == 1 {
6      return 1
7    }
8
9    for n >= 3 {
10     n--
11     f = f1 + f2
12     f1 = f2
13     f2 = f
14   }
15   return f
16 }
```

Listing 5:

```
1  func f_iter2(n int) uint64 {
2    var f uint64
3    var f1, f2 uint64 = 1, 1
4
5    if n == 2 || n == 1 {
6      return 1
7    }
8
9    for i := 2; i <= n; i++ {
10     f = f1 + f2
11     f1 = f2
12     f2 = f
13   }
14   return f
15 }
```

Senza eseguire le funzioni al computer, rispondete alle seguenti domande:

4. Considerando solo il valore restituito, le due funzioni sono equivalenti? (ovvero: restituiscono sempre lo stesso valore?)
5. Le due funzioni sono equivalenti alla funzione `f_rec`?
6. Modificate (se necessario) le funzioni `f_iter1` e `f_iter2` in modo che risultino essere equivalenti a `f_rec`.
7. Stimate il numero di operazioni che si svolgono durante l'esecuzione di `f_rec`, `f_iter1` e `f_iter2`: sono paragonabili?

Considerate infine la seguente funzione ricorsiva `f_riter`

Listing 6:

```
1  func f_riter(a, b uint64, n int) uint64 {
2    if n == 2 {
3      return a
4    }
5
6    if n == 1 {
7      return b
8    }
9
10   return f_riter(a+b, a, n-1)
11 }
```

Senza eseguire la funzione al computer, rispondete alle seguenti domande:

8. Convincetevi che questa funzione può essere usata per calcolare `f_rec`. In particolare: con quali argomenti devo invocare `f_riter` per ottenere il valore restituito da `f_rec(n)`?
9. Rappresentate graficamente lo schema delle chiamate ricorsive definiti dall'invocazione `f_rec(7)` e dalla chiamata equivalente del tipo `f_riter(..., ..., ...)`.
10. Considerate il numero di chiamate ricorsive effettute da `f_rec(n)` e dalla chiamata equivalente del tipo `f_riter(..., ..., ...)`. Sono paragonabili?

11. Usate una variabile globale `counter` per tenere traccia del numero delle chiamate ricorsive; quindi scrivete un programma che invoca `f_rec` e `f_riter` e stampa, oltre al valore restituito, anche il numero di chiamate della funzione.

Una volta concluso l'esercizio potete usare il file `rec-iter.go` per fare degli esperimenti. Analizzate il codice per capire come usarlo e come interpretarne l'output!

5 Torri di Hanoi

Il gioco delle torri di Hanoi (anche detto delle torri di Brahma) è stato inventato nel 1883 dal matematico francese Edouard Lucas. L'obiettivo è spostare da un paletto ad un altro un certo numero di dischi forati di dimensione crescente infilati sul paletto e appoggiati l'uno sull'altro. Le regole del gioco sono che si può spostare solo il disco che è in cima ad una pila e non si deve mai appoggiare un disco di dimensione più grande sopra uno più piccolo; per aiutarsi, è possibile usare un terzo paletto come appoggio ausiliario. Secondo una leggenda (forse inventata dal matematico stesso) alcuni monaci di un tempio Indù sono costantemente impegnati a spostare sessantaquattro dischi secondo le regole del gioco; la leggenda dice che quando i monaci completeranno il lavoro, il mondo finirà!

Obiettivo dell'esercizio è scrivere un programma in grado di giocare al gioco delle torri di Hanoi, ossia di specificare la sequenza di mosse da effettuare per risolvere il rompicapo data l'altezza $n > 0$ della pila. Potete assumere che i tre paletti siano numerati da 0 a 2 e che gli n dischi siano inizialmente impilati dal più piccolo (in cima alla pila) al più grande (sotto tutta la pila) sul paletto 0 e vadano spostati al paletto 2. Per semplicità, le mosse sono date semplicemente dall'indicazione del paletto da e verso cui si deve muovere il disco.

Ad esempio, una soluzione per una pila di altezza 3 è data dalla seguente sequenza di mosse:

```
0 -> 2
0 -> 1
2 -> 1
0 -> 2
1 -> 0
1 -> 2
0 -> 2
```

Questo vuol dire che il disco più piccolo va spostato dal paletto 0 al paletto 2, quindi il disco mediano, ora in cima al paletto 0, va spostato al paletto 1; a questo punto il disco più piccolo (rimasto sul paletto 2) va rimesso sopra il mediano (ora sul paletto 1) e, finalmente, il disco più grande va spostato dal paletto 0 al paletto 2, nella sua posizione finale. Le restanti tre mosse, spostando i due dischi rimanenti dal paletto 1 al paletto 2.

Scrivete una funzione `hanoi( n , from, temp, to int )` che stampi le mosse per spostare `n` dischi dal paletto `from` al paletto `to` aiutandosi, se necessario, con il paletto ausiliario `temp`. Osservate che tale funzione, posto che siano state stampate le mosse per spostare $n-1$ dischi da `from` a `temp` (usando eventualmente `to` come paletto ausiliario) può stampare la mossa `from ->to` e quindi stampare le rimanenti mosse necessarie a spostare gli $n-1$ dischi da `temp` a `to`

(usando eventualmente `from` come paletto ausiliario). Questa osservazione dovrebbe suggerirvi immediatamente una implementazione ricorsiva della funzione `hanoi`.

La fine del mondo. Modificate la funzione precedente perché restituisca soltanto il numero di mosse effettuate (invece che stamparle). Come cresce tale numero al crescere del numero di dischi? Per rendervi meglio conto del tasso di crescita, provate a considerare il logaritmo del numero di mosse, come cresce?

Vi sembra realistico che per spostare 64 dischi ci voglia un tempo pari alla durata del mondo?

Le torri in dettaglio Supponendo ora di chiamare i dischi dal più grande al più piccolo con le lettere `A`, `B`, `C`, ..., scrivete ora una versione più dettagliata della funzione appena sviluppata che stampi ad ogni passo del gioco il contenuto di ogni paletto (usando una linea per ogni passo e separando il contenuto dei tre paletti con una virgola). Ad esempio, se la pila iniziale ha altezza 3, ed è quindi data da `ABC`, la nuova funzione deve scrivere le seguenti mosse

```
ABC, ,
AB, , C
A, B, C
A, BC,
, BC, A
C, B, A
C, , AB
, , ABC
```