# Laboratorio di algoritmi e strutture dati

Ancora sugli alberi*

Docente: Violetta Lonati

## 1 Dipendenti

Nella multinazionale Algoré il lavoro è organizzato in maniera gerarchica. Ogni dipendente è inquadrato in un certo *livello di impiego*. Tranne i dipendenti di *massimo livello*, ogni dipendente ha un *supervisore*, di cui è detto *subordinato*. Un dipendente può avere 0, 1, o più subordinati.

Si considerino i seguenti compiti.

(a) Dato un certo dipendente, stampare l'elenco dei suoi subordinati.

(b) Contare quanti sono i dipendenti che non hanno alcun subordinato.

(c) Dato un certo dipendente, individuare chi è il suo supervisore.

(d) Dato un certo dipendente, stampare la lista dei dipendenti che si trovano sopra di lui gerarchicamente, partendo dal suo supervisore e risalendo la gerarchia fino a un dipendente di massimo livello.

(e) Stampare l'elenco di tutti i dipendenti –non importa l'ordine–, indicando per ciascuno chi è il suo supervisore (tranne nel caso di dipendenti di massimo livello).

**Esempio** Anna è supervisore di Bruno, Carlo e Daniela. Bruno è supervisore di Enrico e Francesco. Gianni è supervisore di Harry. Francesco è supervisore di Irene. Il numero di dipendenti senza subordinati è 5 (Carlo, Daniela, Enrico, Harry, Irene). La lista di dipendenti che si trovano sopra Irene è: Francesco, Bruno, Anna. Questo è l'elenco dei dipendenti in ordine di livello: A, G (massimo livello), B, D, H, C (subordinati di dipendenti di massimo livello, non importa il loro ordine), F, E, I.

*Ultimo aggiornamento: 17 novembre 2022 - 07:31:43

### 1.1 Modellazione e progettazione

1. *Modellate la situazione* con una struttura dati opportuna:
   - *descrivete* come si possono rappresentare i dipendenti e le loro relazioni con la struttura dati scelta;
   - *riformulate*, nei termini della struttura dati scelta, ciascuno dei *compiti* enunciati sopra.
2. Descrivete come è opportuno *implementare la struttura dati scelta*.
3. Per ciascun compito, *progettate e descrivete un algoritmo* che consente di svolgere il compito, sfruttando le scelte di progettazione e implementazione fatte precedentemente. Gli algoritmi possono essere descritti a parole o in pseudocodice; può essere opportuno fare riferimento ad algoritmi noti.

### 1.2 Implementazione

Definite, in Go, uno o più tipi di dati utili a rappresentare i dipendenti e le loro relazioni, in base alle scelte fatte ai punti precedenti.

Scrivete quindi una funzione Go per ciascuno dei compiti enunciati sopra (potete scrivere funzioni ausiliarie se lo ritenete opportuno). Le funzioni devono avere questi nomi:
(a) `stampaSubordinati`
(b) `quantiSenzaSubordinati`
(c) `supervisore`
(d) `stampaImpiegatiSopra`
(e) `stampaImpiegatiConSupervisore`
Scrivete anche un programma per testare le funzioni.

## 2 L'aritmetica dei pesci

Questo esercizio è ispirato alla sfida proposta da Advent of Code nel 2021, giorno 18[1].

Lo scopo di questa traccia è guidarvi nella costruzione di una soluzione alla sfida: la descrizione del compito è intervallata da commenti e consegne intermedie, da svolgere man mano che si progredisce con la lettura.

Se preferite, potete provare a risolvere la sfida per conto vostro, partendo dalla sfida originale proposta sul sito di Advent of Code; confrontante poi la vostra soluzione con quanto proposto qui. Naturalmente non c'è un solo modo per risolvere il problema: in questa scheda vi propongo un approccio basato sugli alberi.

### 2.1 I numeri secondo i pesci

I pesci fanno i conti in un modo particolare. Tanto per cominciare, i numeri dei pesci non sono numeri normali. Per i pesci, un numero è una coppia formata da due elementi. Ciascun elemento

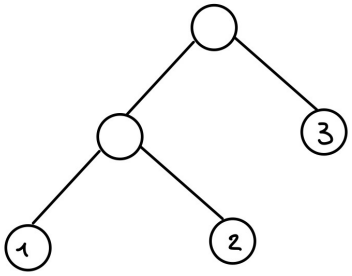
[1] `https://adventofcode.com/2021/day/18`

della coppia può essere un numero intero oppure un'altra coppia. Chiameremo questi numeri *numeri-pesce*. I numeri-pesce sono scritte usando le parentesi: `[x,y]` indica la coppia formata dagli elementi `x` e `y`.

**Esempio.** Ecco alcuni esempi di numeri-pesce:

```
[1,2]
[[1,2],3]
[9,[8,7]]
[[1,9],[8,5]]
[[[[1,2],[3,4]],[[5,6],[7,8]]],9]
[[[9,[3,8]],[[0,9],6]],[[[3,7],[4,9]],3]]
[[[[1,3],[5,3]],[[1,3],[8,7]]],[[[4,9],[6,9]],[[8,2],[7,3]]]]
```

**Commento.** I numeri-pesce possono essere rappresentati come alberi. Ad esempio il secondo numero qui sopra può essere rappresentato così:




**Consegna.** Disegnate anche gli altri alberi corrispondenti ai numeri-pesce dell'esempio. Di che tipo di alberi si tratta? Cosa contengono i nodi di questi alberi? C'è differenza tra foglie, nodi interni e radice? Come si può caratterizzare la profondità dei nodi? Provate a dare una descrizione a parole di come passare da un numero-pesce a un albero e viceversa

**Consegna.**

- Definite un tipo di dato di Go per rappresentare i numeri-pesci come alberi.
- Scrivete una funzione `readTree` che, data una stringa con parentesi che indica un numero-pesce, costruisca l'albero corrispondente.
- Scrivete una funzione `printTree` che, dato un albero che rappresenta un numero-pesce, stampi il numero-pesce (come stringa con parentesi).
- Scrivete una funzione `treeToSlice` che, dato un albero che rappresenta un numero-pesce, produce una slice con le sue foglie, visitate da sinistra a destra. Ad esempio, sul numero-pesce rappresentato graficamente qui sopra, la slice conterrà nell'ordine i tre nodi 1,2,3.
- Testate le funzioni precedenti su tutti gli esempi sopra.

## 2.2 Somma di numeri-pesce

La somma di due numeri-pesce `x` e `y` è data dalla coppia `[x,y]`. Ad esempio, `[1,2] + [[3,4],5]` vale `[[1,2],[[3,4],5]]`.

**Consegna** Scrivete una funzione che, dati due alberi rappresentanti due numeri-pesce, restituisce l'albero che corrisponde alla loro somma.

Tutti i numeri-pesce devono essere in *forma ridotta*. Questo si rende necessario in particolare con i numeri risultanti dalla somma di numeri-pesce.

La riduzione dei pesci avviene ripetendo la prima azione che compare in questa lista

```
Se c'è qualche coppia annidata dentro 4 coppie,
  l'elemento di sinistra della prima di tali coppie \emph{esplode}.
Se c'è qualche elemento (numero regolare) di un numero-pesce più grande di 10,
  il più a sinistra di questi numeri maggiori di 10 viene \emph{suddiviso}
Se non si può applicare nessuna delle due operazioni precedenti,
  il numero-pesce è ridotto.
```

Per ridurre un numero-pesce è utile manipolare la sua rappresentazione "sequenziale" in forma di slice. Poiché la slice contiene puntatori a nodi, se si modificano i collegamenti interni di questi nodi, si modifica anche l'albero

Per individuare il numero regolare subito a destra e il numero regolare subito a sinistra di una coppia, può essere utile "sequenziare" l'albero, ovvero vederlo come una sequenza di nodi. A questo fine, è utile usare una slice di supporto in cui memorizzare, in ordine, i puntatori ai nodi dell'albero. Per farlo, si può fare una visita dell'albero stesso.

**Consegna** Definite le segnature di due funzioni `explode` e `split` che realizzano le operazioni di esplosione e di suddivisione. E' opportuno passare come argomento di queste funzioni il numero-pesce sequenziato come slice di puntatori a nodi, invece dell'albero in sé. Impostate quindi una funzione `reduce` che, dati un alberi rappresentante un numero-pesce, restituisce l'albero che corrisponde alla sua forma ridotta, sfruttando le due funzioni precedente (per testare la funzione `reduce`, bisogna aspettare di aver implementato le due funzioni di esplosione e suddivisione)

## 2.3 Esplosione

In base alle regole sopra, si può essere sicuri che quando c'è una coppia da esplodere, questa sarà formata da una coppia di numeri. Per esplodere una coppia, il valore di sinistra è aggiunto al primo numero regolare a sinistra della coppia esplosa (se c'è), e il valore destro è aggiunto al primo numero regolare a destra della coppia (se c'è). Poi, l'intera coppia è rimpiazzata dal numero regolare 0.

**Esempi** Ecco alcuni esempi di esplosione

- `[[[[[9,8],1],2],3],4]` diventa `[[[[0,9],2],3],4]`: viene esplosa la coppia `[9,8]` e `9` non ha nessun numero a sinistra.

- `[7,[6,[5,[4,[3,2]]]]]` diventa `[7,[6,[5,[7,0]]]]` (: viene esplosa la coppia `[3,2]` e `2` non ha nessun numero a destra.
- `[[6,[5,[4,[3,2]]]],1]` diventa `[[6,[5,[7,0]]],3]`: viene esplosa la coppia `[3,2]`
- `[[3,[2,[1,[7,3]]]],[6,[5,[4,[3,2]]]]]` diventa `[[3,[2,[8,0]]],[9,[5,[4,[3,2]]]]]`: viene esplosa la coppia `[7,3]`, che è la coppia dentro 4 parentesi, più a sinistra
- `[[3,[2,[8,0]]],[9,[5,[4,[3,2]]]]]` diventa `[[3,[2,[8,0]]],[9,[5,[7,0]]]]`

**Consegna** Disegnate i numeri-pesce dell'esempio precedente come alberi. Descrivete l'operazione di esplosione in termini di alberi. Che caratteristiche hanno gli alberi esplosi rispetto a quelli di partenza?

**Consegna** Implementate la funzione `explode`.