Fondamenti di Informatica (Elettronici)

THINK JULIA – Chapter 13

25 novembre 2020

13. Case Study: Data Structure Selection[1]

1. Analisi della frequenza delle parole
2. Random Numbers
3. Istogramma di parole
4. Parole più comuni
5. Parametri opzionali
6. Sottrazione di dizionari
7. Parole casuali
8. Strutture dati
9. Debugging
10. Glossary
11. Exercises

[1] From https://benlauwens.github.io/ThinkJulia.jl/latest/book.html. It is available under the Creative Commons Attribution-NonCommercial 3.0 Unported License.

A questo punto hai imparato a conoscere le strutture dati principali di Julia e hai visto alcuni degli algoritmi che le utilizzano.

Questo capitolo presenta un caso di studio con esercizi che consentono di pensare alla scelta delle strutture di dati e di esercitarsi ad usarle.

Section 1

Analisi della frequenza delle parole

Analisi della frequenza delle parole 1/3

Come al solito, dovresti almeno provare gli esercizi prima di leggere le soluzioni.

Analisi della frequenza delle parole 1/3

Come al solito, dovresti almeno provare gli esercizi prima di leggere le soluzioni.

Esercizio 13-1

Scrivi un programma che:

- legge un file,

Analisi della frequenza delle parole 1/3

Come al solito, dovresti almeno provare gli esercizi prima di leggere le soluzioni.

Esercizio 13-1

Scrivi un programma che:

- legge un file,
- spezza ogni riga in parole,

Analisi della frequenza delle parole 1/3

Come al solito, dovresti almeno provare gli esercizi prima di leggere le soluzioni.

Esercizio 13-1

Scrivi un programma che:

- legge un file,
- spezza ogni riga in parole,
- elimina gli spazi bianchi e la punteggiatura dalle parole e

Analisi della frequenza delle parole 1/3

Come al solito, dovresti almeno provare gli esercizi prima di leggere le soluzioni.

Esercizio 13-1

Scrivi un programma che:

- legge un file,
- spezza ogni riga in parole,
- elimina gli spazi bianchi e la punteggiatura dalle parole e
- li converte in minuscolo.

Analisi della frequenza delle parole 1/3

Come al solito, dovresti almeno provare gli esercizi prima di leggere le soluzioni.

Esercizio 13-1

Scrivi un programma che:

- legge un file,
- spezza ogni riga in parole,
- elimina gli spazi bianchi e la punteggiatura dalle parole e
- li converte in minuscolo.

Analisi della frequenza delle parole 1/3

Come al solito, dovresti almeno provare gli esercizi prima di leggere le soluzioni.

Esercizio 13-1

Scrivi un programma che:

- legge un file,
- spezza ogni riga in parole,
- elimina gli spazi bianchi e la punteggiatura dalle parole e
- li converte in minuscolo.

MANCIA

La funzione `isletter` verifica se un carattere è alfabetico.

Analisi della frequenza delle parole 2/3

Esercizio 13-2

- Vai a Project Gutenberg e scarica il tuo libro preferito non protetto da copyright in formato testo.

Analisi della frequenza delle parole 2/3

Esercizio 13-2

- Vai a Project Gutenberg e scarica il tuo libro preferito non protetto da copyright in formato testo.
- Modifica il programma dell'esercizio precedente per leggere il libro che hai scaricato, salta le informazioni di intestazione all'inizio del file ed elabora il resto delle parole come prima.

Analisi della frequenza delle parole 2/3

Esercizio 13-2

- Vai a Project Gutenberg e scarica il tuo libro preferito non protetto da copyright in formato testo.
- Modifica il programma dell'esercizio precedente per leggere il libro che hai scaricato, salta le informazioni di intestazione all'inizio del file ed elabora il resto delle parole come prima.
- Quindi modifica il programma per contare il numero totale di parole nel libro e il numero di volte in cui ogni parola viene utilizzata.

Analisi della frequenza delle parole 2/3

Esercizio 13-2

- Vai a Project Gutenberg e scarica il tuo libro preferito non protetto da copyright in formato testo.
- Modifica il programma dell'esercizio precedente per leggere il libro che hai scaricato, salta le informazioni di intestazione all'inizio del file ed elabora il resto delle parole come prima.
- Quindi modifica il programma per contare il numero totale di parole nel libro e il numero di volte in cui ogni parola viene utilizzata.
- Stampa il numero di parole diverse usate nel libro.

Analisi della frequenza delle parole 2/3

Esercizio 13-2

- Vai a Project Gutenberg e scarica il tuo libro preferito non protetto da copyright in formato testo.
- Modifica il programma dell'esercizio precedente per leggere il libro che hai scaricato, salta le informazioni di intestazione all'inizio del file ed elabora il resto delle parole come prima.
- Quindi modifica il programma per contare il numero totale di parole nel libro e il numero di volte in cui ogni parola viene utilizzata.
- Stampa il numero di parole diverse usate nel libro.
- Confronta libri diversi di autori diversi, scritti in epoche diverse.

Analisi della frequenza delle parole 2/3

Esercizio 13-2

- Vai a Project Gutenberg e scarica il tuo libro preferito non protetto da copyright in formato testo.
- Modifica il programma dell'esercizio precedente per leggere il libro che hai scaricato, salta le informazioni di intestazione all'inizio del file ed elabora il resto delle parole come prima.
- Quindi modifica il programma per contare il numero totale di parole nel libro e il numero di volte in cui ogni parola viene utilizzata.
- Stampa il numero di parole diverse usate nel libro.
- Confronta libri diversi di autori diversi, scritti in epoche diverse.
- Quale autore utilizza il vocabolario più ampio?

Analisi della frequenza delle parole 3/3

Esercizio 13-3

Modifica il programma dell'esercizio precedente per stampare le 20 parole usate più di frequente nel libro.

Esercizio 13-4

Modificare il programma precedente per leggere un elenco di parole e quindi stampare tutte le parole del libro che non sono nell'elenco di parole.
Quanti di loro sono errori di battitura?
Quante di loro sono parole comuni che dovrebbero essere nella lista delle parole e quante sono davvero oscure?

Section 2

Random Numbers

Calcolo deterministico

Dati gli stessi input, la maggior parte dei programmi per computer genera ogni volta gli stessi output, quindi si dice che siano deterministici.

- Il determinismo di solito è una buona cosa, poiché ci aspettiamo che lo stesso calcolo produca lo stesso risultato.

Calcolo deterministico

Dati gli stessi input, la maggior parte dei programmi per computer genera ogni volta gli stessi output, quindi si dice che siano deterministici.

- Il determinismo di solito è una buona cosa, poiché ci aspettiamo che lo stesso calcolo produca lo stesso risultato.
- Per alcune applicazioni, tuttavia, vogliamo che il computer sia imprevedibile.

Calcolo deterministico

Dati gli stessi input, la maggior parte dei programmi per computer genera ogni volta gli stessi output, quindi si dice che siano deterministici.

- Il determinismo di solito è una buona cosa, poiché ci aspettiamo che lo stesso calcolo produca lo stesso risultato.
- Per alcune applicazioni, tuttavia, vogliamo che il computer sia imprevedibile.
- I giochi sono un esempio ovvio, ma ce ne sono di più.

Calcolo deterministico

Dati gli stessi input, la maggior parte dei programmi per computer genera ogni volta gli stessi output, quindi si dice che siano deterministici.

- Il determinismo di solito è una buona cosa, poiché ci aspettiamo che lo stesso calcolo produca lo stesso risultato.
- Per alcune applicazioni, tuttavia, vogliamo che il computer sia imprevedibile.
- I giochi sono un esempio ovvio, ma ce ne sono di più.
- Rendere un programma veramente non deterministico risulta essere difficile, ma ci sono modi per farlo sembrare almeno non deterministico.

Calcolo deterministico

Dati gli stessi input, la maggior parte dei programmi per computer genera ogni volta gli stessi output, quindi si dice che siano deterministici.

- Il determinismo di solito è una buona cosa, poiché ci aspettiamo che lo stesso calcolo produca lo stesso risultato.
- Per alcune applicazioni, tuttavia, vogliamo che il computer sia imprevedibile.
- I giochi sono un esempio ovvio, ma ce ne sono di più.
- Rendere un programma veramente non deterministico risulta essere difficile, ma ci sono modi per farlo sembrare almeno non deterministico.
- Uno di questi è utilizzare algoritmi che generano** numeri pseudocasuali**.

Calcolo deterministico

Dati gli stessi input, la maggior parte dei programmi per computer genera ogni volta gli stessi output, quindi si dice che siano deterministici.

- Il determinismo di solito è una buona cosa, poiché ci aspettiamo che lo stesso calcolo produca lo stesso risultato.
- Per alcune applicazioni, tuttavia, vogliamo che il computer sia imprevedibile.
- I giochi sono un esempio ovvio, ma ce ne sono di più.
- Rendere un programma veramente non deterministico risulta essere difficile, ma ci sono modi per farlo sembrare almeno non deterministico.
- Uno di questi è utilizzare algoritmi che generano** numeri pseudocasuali**.
- I numeri pseudocasuali non sono veramente casuali perché sono generati da un calcolo deterministico, ma solo guardando i numeri è

Funzione rand

La funzione rand restituisce un float casuale compreso tra 0.0 e 1.0 (incluso 0.0 ma non 1.0).

Ogni volta che chiami rand, ottieni il numero successivo di una lunga serie. Per vedere un esempio, esegui questo ciclo:

```
for i in 1:10
    x = rand()
    println(x)
end
```

La funzione rand può prendere un iteratore o un array come argomento e restituisce un elemento casuale:

```
for i in 1:10
    x = rand(1:6)
    print(x, " ")
end
```

Exercise 13-5

Scrivi una funzione denominata `choosefromhist` che prenda un istogramma come definito nel Dizionario come una raccolta di contatori e restituisca un valore casuale dall'istogramma, scelto con probabilità proporzionale alla frequenza.

Ad esempio, per questo istogramma:

```
julia> t = ['a', 'a', 'b'];
julia> histogram(t)
Dict{Any,Any} with 2 entries:
    'a' => 2
    'b' => 1
```

La tua funzione dovrebbe restituire `a` con probabilità $\frac{2}{3}$ e `b` con probabilità $\frac{1}{3}$.

Section 3

Istogramma di parole

Elaborare un file e una linea

Dovresti provare gli esercizi precedenti prima di andare avanti.

Avrai anche bisogno di *
https://github.com/BenLauwens/ThinkJulia.jl/blob/master/data/emma.txt*.

Ecco un programma che legge un file e costruisce un istogramma delle parole nel file:

```
function processfile(filename)
    hist = Dict()
    for line in eachline(filename)
        processline(line, hist)
    end
    hist
end;

function processline(line, hist)
    line = replace(line, '-' => ' ')
    for word in split(line)
        word = string(filter(isletter, [word...])...)
        word = lowercase(word)
        hist[word] = get!(hist, word, 0) + 1
    end
end;
```

aaaa

```
hist = processfile("emma.txt");
```

aaaa

```
hist = processfile("emma.txt");
```

- Questo programma legge `emma.txt`, che contiene il testo di Emma di Jane Austen.

aaaa

```
hist = processfile("emma.txt");
```

- Questo programma legge `emma.txt`, che contiene il testo di Emma di Jane Austen.
- `processfile` scorre le righe del file, passandole una alla volta a `processline`.

aaaa

```
hist = processfile("emma.txt");
```

- Questo programma legge `emma.txt`, che contiene il testo di Emma di Jane Austen.
- `processfile` scorre le righe del file, passandole una alla volta a `processline`.
- L'istogramma `hist` viene utilizzato come accumulatore.

aaaa

```
hist = processfile("emma.txt");
```

- Questo programma legge `emma.txt`, che contiene il testo di Emma di Jane Austen.
- `processfile` scorre le righe del file, passandole una alla volta a `processline`.
- L'istogramma `hist` viene utilizzato come accumulatore.
- `processline` usa la funzione `replace` per sostituire i trattini con spazi prima di usare `split` per spezzare la `line` in un array di stringhe.

aaaa

```
hist = processfile("emma.txt");
```

- Questo programma legge `emma.txt`, che contiene il testo di Emma di Jane Austen.
- `processfile` scorre le righe del file, passandole una alla volta a `processline`.
- L'istogramma `hist` viene utilizzato come accumulatore.
- `processline` usa la funzione `replace` per sostituire i trattini con spazi prima di usare `split` per spezzare la `line` in un array di stringhe.
- Attraversa l'array di parole e usa `filter`, `isletter` e `lowercase` per rimuovere la punteggiatura e convertirle in minuscolo.

aaaa

```
hist = processfile("emma.txt");
```

- Questo programma legge `emma.txt`, che contiene il testo di Emma di Jane Austen.
- `processfile` scorre le righe del file, passandole una alla volta a `processline`.
- L'istogramma `hist` viene utilizzato come accumulatore.
- `processline` usa la funzione `replace` per sostituire i trattini con spazi prima di usare `split` per spezzare la `line` in un array di stringhe.
- Attraversa l'array di parole e usa `filter`, `isletter` e `lowercase` per rimuovere la punteggiatura e convertirle in minuscolo.
- (È un'abbreviazione dire che le stringhe vengono "convertite"; ricorda che le stringhe sono "immutabili", quindi una funzione come `lowercase` restituisce nuove stringhe.)

aaaa

```
hist = processfile("emma.txt");
```

- Questo programma legge `emma.txt`, che contiene il testo di Emma di Jane Austen.
- `processfile` scorre le righe del file, passandole una alla volta a `processline`.
- L'istogramma `hist` viene utilizzato come accumulatore.
- `processline` usa la funzione `replace` per sostituire i trattini con spazi prima di usare `split` per spezzare la `line` in un array di stringhe.
- Attraversa l'array di parole e usa `filter`, `isletter` e `lowercase` per rimuovere la punteggiatura e convertirle in minuscolo.
- (È un'abbreviazione dire che le stringhe vengono "convertite"; ricorda che le stringhe sono "immutabili", quindi una funzione come `lowercase` restituisce nuove stringhe.)
- Infine, `processline` aggiorna l'istogramma creando un nuovo elemento o incrementandone uno esistente.

Contare il numero totale di parole

Per** contare il numero totale** di parole nel file, possiamo sommare le frequenze nell'istogramma:

```
function totalwords(hist)
    sum(values(hist))
end
```

Il numero di parole diverse è solo il numero di elementi nel dizionario:

```
function differentwords(hist)
    length(hist)
end
```

Ecco del codice per stampare i risultati:

```
julia> println("Total number of words: ", totalwords(hist))
Total number of words: 162742
julia> println("Number of different words: ", differentwords(hist))
Number of different words: 7380
```

Section 4

Parole più comuni

Stampa le 10 parole più comuni 1/2

Per trovare le parole più comuni, possiamo creare un array di tuple, in cui ogni tupla contiene una parola e la sua frequenza, e ordinarla.

La seguente funzione prende un istogramma e restituisce un array di tuple `word-frequency`:

```
function mostcommon(hist)
    t = []
    for (key, value) in hist
        push!(t, (value, key))
    end
    reverse(sort(t))
end
```

In ogni tupla, la frequenza viene visualizzata per prima, quindi la matrice risultante viene ordinata per frequenza. Ecco un ciclo che stampa le 10 parole più comuni:

```
t = mostcommon(hist)
println("The most common words are:")
for (freq, word) in t[1:10]
    println(word, "\t", freq)
```

Stampa le 10 parole più comuni 2/2

Uso un carattere di tabulazione (`\t`) come "separatore", piuttosto che uno spazio, quindi la seconda colonna è allineata. Ecco i risultati di Emma:

```
The most common words are:
to  5295
the 5266
and 4931
of  4339
i   3191
a   3155
it  2546
her 2483
was 2400
she 2364
```

Questo codice può essere semplificato usando l'argomento della parola chiave `rev` (reverse) della funzione `sort`. Puoi leggere a riguardo su https://docs.julialang.org/en/v1/base/sort/#Base.sort.

Section 5

Parametri opzionali

Funzioni che accettano argomenti opzionali

Abbiamo visto funzioni predefinite che accettano argomenti opzionali.

È anche possibile scrivere funzioni definite dal programmatore con argomenti opzionali.

Ad esempio, ecco una funzione che stampa le parole più comuni in un istogramma:

```
function printmostcommon(hist, num=10)
    t = mostcommon(hist)
    println("The most common words are: ")
    for (freq, word) in t[1:num]
        println(word, "\t", freq)
    end
end
```

Parametri obbligatori e facoltativi

Il primo parametro è obbligatorio; il secondo è facoltativo. Il valore predefinito di `num` è 10. Se fornisci solo un argomento:

```
printmostcommon (hist)
```

`num` ottiene il valore predefinito. Se fornisci due argomenti:

```
printmostcommon (hist, 20)
```

`num` ottiene invece il valore dell'argomento. In altre parole, l'argomento facoltativo sovrascrive il valore predefinito.

Se una funzione ha entrambi i parametri obbligatori e facoltativi, tutti i parametri obbligatori devono venire prima, seguiti da quelli opzionali.

Section 6

Sottrazione di dizionari

Problema della sottrazione di insiemi

Trovare le parole del libro che non sono nella lista delle parole da `words.txt` è un problema che potresti riconoscere come set subtraction.

Cioè, vogliamo trovare tutte le parole di un insieme (le parole nel libro) che non sono nell'altro (le parole nell'elenco).

`subtract` prende i dizionari `d1` e `d2` e restituisce un nuovo dizionario che contiene tutte le chiavi di `d1` che non sono in `d2`.

Dal momento che non ci interessano davvero i valori, li impostiamo tutti su `nothing`.

```
function subtract(d1, d2)
    res = Dict()
    for key in keys(d1)
        if key !in keys(d2)
            res[key] = nothing
        end
    end
    res
```

Stampa dei risultati 1/2

Per trovare le parole nel libro che non sono in `words.txt`, possiamo usare `processfile` per costruire un istogramma per `words.txt`, e quindi `subtract`:

```
words = processfile("words.txt")
diff = subtract(hist, words)

println("Words in the book that aren't in the word list:")
for word in keys(diff)
    print(word, " ")
end
```

Stampa dei risultati 2/2

Ecco alcuni dei risultati su Emma:

```
Words in the book that aren't in the word list:
outree quicksighted outwardly adelaide rencontre jeffereys
unreserved dixons betweens ...
```

Alcune di queste parole sono nomi e possessivi. Altri, come "`rencontre`", non sono più di uso comune. Ma alcune sono parole comuni che dovrebbero davvero essere nella lista!

Esercizio 13-6

- Julia fornisce una struttura dati chiamata `Set` che fornisce molte operazioni comuni sugli insiemi.

Esercizio 13-6

- Julia fornisce una struttura dati chiamata `Set` che fornisce molte operazioni comuni sugli insiemi.

Esercizio 13-6

- Julia fornisce una struttura dati chiamata `Set` che fornisce molte operazioni comuni sugli insiemi.

Puoi leggere informazioni su di loro in Raccolte e strutture dati o leggere la documentazione all'indirizzo

https://docs.julialang.org/en/v1/base/collections/#Set-Like-Collections-1.

Esercizio 13-6

- Julia fornisce una struttura dati chiamata `Set` che fornisce molte operazioni comuni sugli insiemi.

Puoi leggere informazioni su di loro in Raccolte e strutture dati o leggere la documentazione all'indirizzo

https://docs.julialang.org/en/v1/base/collections/#Set-Like-Collections-1.

- Scrivere un programma che utilizzi set subtraction per trovare le parole nel libro che non sono nell'elenco delle parole.

Section 7

Parole casuali

Parole casuali dall'istogramma

Per scegliere una parola casuale dall'istogramma, l'algoritmo più semplice è costruire un array con più copie di ogni parola, in base alla frequenza osservata, quindi scegliere dall'array:

```
function randomword(h) t = []
    for (word, freq) in h
        for i in 1:freq
            push!(t, word)
        end
    end
    rand(t)
end
```

Questo algoritmo funziona, ma non è molto efficiente; ogni volta che scegli una parola a caso, ricostruisce l' array, che è grande quanto il libro originale.

Un ovvio miglioramento consiste nel creare l'array una volta e quindi effettuare più selezioni, ma l'array è comunque grande.

Una soluzione migliore

Un'alternativa è:

1. Usa `keys` per ottenere un array di parole nel libro.

Esercizio 13-7

Scrivi un programma che utilizzi questo algoritmo per scegliere una parola casuale dal libro.

Una soluzione migliore

Un'alternativa è:

1. Usa `keys` per ottenere un array di parole nel libro.
2. Costruisci un array che contenga la somma cumulativa delle frequenze delle parole (vedi l'Esercizio 10-2). L'ultimo elemento in questo array è il numero "total" di parole nel libro, `n`.

Esercizio 13-7

Scrivi un programma che utilizzi questo algoritmo per scegliere una parola casuale dal libro.

Una soluzione migliore

Un'alternativa è:

1. Usa `keys` per ottenere un array di parole nel libro.
2. Costruisci un array che contenga la somma cumulativa delle frequenze delle parole (vedi l'Esercizio 10-2). L'ultimo elemento in questo array è il numero "total" di parole nel libro, `n`.
3. Scegli un numero casuale da `1` a `n`. Usa una ricerca in bisezione (vedi l'Esercizio 10-10) per trovare l'indice in cui il numero casuale verrebbe inserito nella somma cumulativa.

Esercizio 13-7

Scrivi un programma che utilizzi questo algoritmo per scegliere una parola casuale dal libro.

Una soluzione migliore

Un'alternativa è:

1. Usa `keys` per ottenere un array di parole nel libro.
2. Costruisci un array che contenga la somma cumulativa delle frequenze delle parole (vedi l'Esercizio 10-2). L'ultimo elemento in questo array è il numero "total" di parole nel libro, `n`.
3. Scegli un numero casuale da `1` a `n`. Usa una ricerca in bisezione (vedi l'Esercizio 10-10) per trovare l'indice in cui il numero casuale verrebbe inserito nella somma cumulativa.
4. Usa l'indice per trovare la parola corrispondente nell'array di parole.

Esercizio 13-7

Scrivi un programma che utilizzi questo algoritmo per scegliere una parola casuale dal libro.

Section 8

Strutture dati

aaaa

aaaa

Section 9

Debugging

aaaa

aaaa

Section 10

Glossary

Glossary

deterministic Pertaining to a program that does the same thing each time it runs, given the same inputs.

Glossary

deterministic Pertaining to a program that does the same thing each time it runs, given the same inputs.

pseudorandom Pertaining to a sequence of numbers that appears to be random, but is generated by a deterministic program.

Glossary

deterministic Pertaining to a program that does the same thing each time it runs, given the same inputs.

pseudorandom Pertaining to a sequence of numbers that appears to be random, but is generated by a deterministic program.

default value The value given to an optional parameter if no argument is provided.

Glossary

deterministic Pertaining to a program that does the same thing each time it runs, given the same inputs.

pseudorandom Pertaining to a sequence of numbers that appears to be random, but is generated by a deterministic program.

default value The value given to an optional parameter if no argument is provided.

override To replace a default value with an argument.

Glossary

deterministic Pertaining to a program that does the same thing each time it runs, given the same inputs.

pseudorandom Pertaining to a sequence of numbers that appears to be random, but is generated by a deterministic program.

default value The value given to an optional parameter if no argument is provided.

override To replace a default value with an argument.

benchmarking The process of choosing between data structures by implementing alternatives and testing them on a sample of the possible inputs.

Glossary

deterministic Pertaining to a program that does the same thing each time it runs, given the same inputs.

pseudorandom Pertaining to a sequence of numbers that appears to be random, but is generated by a deterministic program.

default value The value given to an optional parameter if no argument is provided.

override To replace a default value with an argument.

benchmarking The process of choosing between data structures by implementing alternatives and testing them on a sample of the possible inputs.

rubber duck debugging Debugging by explaining your problem to an inanimate object such as a rubber duck. Articulating the problem can help you solve it, even if the rubber duck doesn't know Julia.

aaaa

Section 11

Exercises

aaaa

aaaa