Programmazione dinamica

The Coding DEI – 27/03/2019

Alessio Mazzetto - alessiomazzetto1995@gmail.com

Programmazione dinamica
Panoramica

- Tecnica risolutiva per problemi complessi
- Basato sulla risoluzione di relazioni di ricorrenza
- Diversi approcci implementativi:

1) Top down 2) Bottom up

Sequenza di fibonacci
Warmup

Per ogni $n \geq 0$, definiamo l' n-esimo numero di Fibonacci come $f(n)$:

$$f(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ f(n-1) + f(n-2), & n \geq 2 \end{cases}$$

Sequenza di fibonacci
Warmup

Per ogni $n \geq 0$, definiamo l' n-esimo numero di Fibonacci come $f(n)$:

$$f(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ f(n-1) + f(n-2), & n \geq 2 \end{cases}$$

Caso base

Equazione di ricorrenza

Sequenza di fibonacci
Warmup

$$f(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ f(n-1) + f(n-2), & n \geq 2 \end{cases}$$

Come computare questa relazione di ricorrenza?

Sequenza di fibonacci
Bruteforce

$$f(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ f(n-1) + f(n-2), & n \geq 2 \end{cases}$$

Approccio brute-force: scrivo una funzione ricorsiva che «implementa» direttamente la relazione di ricorrenza

Fibonacci con brute-force: numero di somme per calcolare $f(n)$ si può dimostrare che è maggiore di $f(n)$ per $n \geq 4$
→ complessità computazionale esponenziale!

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Formule di ricorrenza
Definizioni

$$f(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ f(n-1) + f(n-2), & n \geq 2 \end{cases}$$

Stato: l'argomento della funzione $f(\cdot)$.

Ogni stato m è associato a una *soluzione* → $f(m)$

Input N: vogliamo calcolare $f(N)$**.** Dobbiamo risolvere $f(\cdot)$ potenzialmente per tutti gli stati $\mathrm{m},\ 0 \leq m < N$ **(sotto-problemi)**

Formule di ricorrenza
Complessità del problema

$$f(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ f(n-1) + f(n-2), & n \geq 2 \end{cases}$$

Input N: vogliamo calcolare $f(N)$

Potenzialmente dobbiamo calcolare $f(\cdot)$ per $O(N)$ stati.
(in altre parole: dobbiamo risolvere $O(N)$ sottoproblemi)

Se conosco $f(n-2)$ e $f(n-1)$, quanto tempo per calcolare $f(n)$? → $O(1)$, devo fare solo una somma!

Programmazione dinamica
Panoramica

- Voglio calcolare ogni stato una singola volta! La ricorsione brute-force calcola lo stesso stato più volte.

- Due metodi implementativi: Top-down e Bottom-up

- Complessità computazionale (formula generale)

$$O(\#numero\ stati) \cdot O(tempo\ max.calcolo\ soluzione\ di\ uno\ stato)$$

Assumendo di conoscere le soluzioni di ogni stato

Fibonacci: $O(N) \cdot O(1) = O(N)$

Sequenza di fibonacci

Top-down

$$f(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ f(n-1) + f(n-2), & n \geq 2 \end{cases}$$

TOP-DOWN: Implementazione ricorsiva con *memoizzazione*

Calcolato $f(m)$ per un certo stato m, memorizziamo il suo valore. Alla prossima invocazione di $f(m)$, restituiamo direttamente il valore memorizzato.

Sequenza di fibonacci

Bottom-up

$$f(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ f(n-1) + f(n-2), & n \geq 2 \end{cases}$$

BOTTOM-UP: parto dai casi base, e calcolo in ordine $f(2), f(3), \ldots$

Stato:	0	1	2	3	4	5	
$f(\cdot)$	0	1	1	2	3	?	…

Programmazione dinamica

La dieta di Poldo (variante della Longest Increasing Subsequence)

Poldo, amante dei panini, non può mangiare un panino che abbia un peso maggiore o uguale a quello appena mangiato.

A Poldo vengono offerti, in ordine, N panini, ognuno con un suo peso. Poldo può decidere se mangiare o no un panino offerto. Poldo può mangiare un panino se:

1. Il panino è il primo che mangia
2. Il panino non ha un peso maggiore o uguale all'ultimo panino che ha mangiato.

La dieta di poldo
Input/Output

Input:

N**:** numero di panini offerti a Poldo

$w_0, w_1, \dots, w_{N-1}$ *:* I pesi dei panini nell'ordine in cui sono offerti a Poldo.

Output:

Vogliamo calcolare il numero massimo di panini che Poldo può mangiare, scegliendo quali mangiare in modo ottimo.

La dieta di poldo

Input/Output

Input:

8

389 *207 155* ***300 299 170 158 65***

Output:

6

Una sequenza ottima è: 389 300 299 170 158 65

La dieta di poldo
Programmazione Dinamica

Vogliamo risolvere il problema utilizzando la Programmazione Dinamica.

Programmazione dinamica → dobbiamo scrivere una relazione di ricorrenza.

Dobbiamo definire una funzione ricorsiva $f(\cdot)$ che risolva il problema.

Come? Si parte capendo cosa può essere uno *stato.*

La dieta di Poldo
Relazione di ricorrenza

Primo tentativo:

$f(n) \rightarrow$ soluzione ottima nella sequenza di panini

$$w_n, w_{n+1}, \dots, w_{N-1}$$

Come calcolo $f(n)$ in funzione dei suoi sottoproblemi?

La dieta di Poldo
Relazione di ricorrenza

Primo tentativo:

$f(n)$ → soluzione ottima nella sequenza di panini

$$w_n, w_{n+1}, \dots, w_{N-1}$$

Come calcolo $f(n)$ in funzione dei suoi sottoproblemi?

PROBLEMA: Non ho nessuna informazione su quali panini Poldo ha già mangiato

La dieta di Poldo
Relazione di ricorrenza

Secondo tentativo:

$f(n)$ → soluzione ottima nella sequenza di panini $w_{n+1}, w_{n+2}, \dots, w_{N-1}$ sapendo che Poldo ha mangiato il panino di peso w_n

Riusciamo a scrivere una formula di ricorrenza?

La dieta di Poldo
Relazione di ricorrenza

$f(n)$ → soluzione ottima nella sequenza di panini $w_{n+1}, w_{n+2}, \dots, w_{N-1}$ sapendo che Poldo ha mangiato il panino di peso w_n

Dobbiamo capire qual è il panino successivo che Poldo mangia tra $w_{n+1}, w_{n+2}, \dots, w_{N-1}$ nella soluzione ottima.
Non lo sappiamo → proviamo tutti quelli possibili!

La dieta di Poldo
Relazione di ricorrenza

$f(n)$ → soluzione ottima nella sequenza di panini $w_{n+1}, w_{n+2}, \dots, w_{N-1}$ sapendo che Poldo ha mangiato il panino di peso w_n

Per $0 \leq n \leq N-1$

$$f(n) = \max_{\substack{i=n+1,\dots,N-1 \\ w_i < w_n}} \{1 + f(i)\} \quad \text{con } \max \emptyset = 0$$

Provo ad accettare qualunque panino in $w_{n+1}, w_{n+2}, \dots, w_{N-1}$ che Poldo può mangiare seguendo la dieta e vedo quale porta alla soluzione migliore (usando il max)

La dieta di Poldo
Relazione di ricorrenza

$f(n)$ → soluzione ottima nella sequenza di panini $w_{n+1}, w_{n+2}, \ldots, w_{N-1}$ sapendo che Poldo ha mangiato il panino di peso w_n

$$f(n) = \max_{\substack{\mathrm{i=n+1,\ldots,N-1} \\ w_i < w_n}} \{1 + f(i)\} \quad \text{con} \quad \max \emptyset = 0$$

La dieta di Poldo

Dove è il caso base?

$$f(n) = \max_{\substack{\mathrm{i=n+1,\dots,N-1} \\ w_i < w_n}} \{1 + f(i)\} \quad \text{con} \quad \max \emptyset = 0$$

$$f(n) = \begin{cases} 0, & \nexists i : n < i < N \land w_i < w_n \\ \max\limits_{\substack{i=n+1,\dots,N-1 \\ w_i < w_n}} \{1 + f(i)\}, & else \end{cases}$$

La dieta di Poldo
Relazione di ricorrenza

$$f(n) = \max_{\substack{\mathrm{i=n+1,\dots,N-1} \\ w_i < w_n}} \{1 + f(i)\} \quad \text{con} \quad \max \emptyset = 0$$

Sappiamo come computare *f(n)*, possiamo usare la programmazione dinamica!

Numero di stati: $O(N)$
Tempo per computare uno stato: $O(N)$ (max di O(N) elementi)
Programmazione dinamica, complessità: $O(N) \cdot O(N) = O(N^2)$

La dieta di Poldo

Relazione di ricorrenza

$$f(n) = \max_{\substack{\mathrm{i=n+1,\dots,N-1} \\ w_i < w_n}} \{1 + f(i)\} \quad \text{con} \quad \max \emptyset = 0$$

Qual è la soluzione al problema? Domanda importante!

Non è $f(0)$, non stiamo contando il primo panino che Poldo mangia. Non è neanche $1 + f(0)$, perché Poldo potrebbe non mangiare, nella soluzione ottima, il panino 0.

La dieta di Poldo
Relazione di ricorrenza

$$f(n) = \max_{\substack{\mathrm{i=n+1,\dots,N-1} \\ w_i < w_n}} \{1 + f(i)\} \quad \text{con} \quad \max \emptyset = 0$$

La soluzione ottima inizierà con Poldo che mangia un panino.

Quale? Non lo sappiamo, li proviamo tutti!

$$answer = \max_{i=0}^{N-1} f(i) + 1$$

La dieta di Poldo
Relazione di ricorrenza

$$f(n) = \max_{\substack{\mathrm{i=n+1,\dots,N-1} \\ w_i < w_n}} \{1 + f(i)\} \quad \text{con} \quad \max \emptyset = 0$$

$$answer = \max_{i=0}^{N-1} f(i) + 1$$

Ora possiamo usare la programmazione dinamica per computare la relazione di ricorrenza e risolvere dunque il problema.

Coin change
Programmazione dinamica

Sia $C = \{c_0, \dots, c_{M-1}\}$ un insieme di valori interi di monete.
Per esempio $C = \{1,\ 3,\ 8,\ 11,\ 23\}$.

Vi viene dato in input un intero *N*

Problema 1: qual è il minimo numero di monete con valore in *C* che sommano a *N.*

Problema 2: in quanti modi distinti posso ottenere complessivamente *N* usando monete con valore in C

Coin change – Problema 1
Programmazione dinamica

Input: *N*, *M* e $C = \{c_0, \dots, c_{M-1}\}$ (tutti interi)
Output: il minimo numero di monete con valore in C da utilizzare per ottenere *N*.

Ex: N=10, M = 3, $C = \{1, 5, 8\}$
La soluzione ottima è 2, infatti 10 = 5+5

Nota bene: la soluzione greedy che prende sempre la moneta con valore piu' grande possibile non funziona in questo caso.

(Fun fact: greedy funziona se C è l'insieme dei «tagli» italiani)
→ Se siete interessati: https://open.kattis.com/problems/canonical

Coin change – Problema 2
Programmazione dinamica

Input: *N*, *M* e $C = \{c_0, \dots, c_{M-1}\}$ (tutti interi)
Output: il numero distinto di modi in cui posso ottenere *N* usando monete con valore in *C*.

Ex: N=8, M = 3, $C = \{1, 2, 5\}$
La soluzione è 7. Questi sono tutti i modi per ottenere 8:

1+1+1+1+1+1+1+1
2+1+1+1+1+1
2+2+1+1+1+1
2+2+2+1+1
2+2+2+2
5+1+1+1
5+2+1

Link alle slide:

https://github.com/yokola95/TCD270319

Nella repository sono presenti anche i codici delle soluzioni in .cpp dei problemi presentati.

Link esterni

Per testare le soluzioni

Poldo:
https://training.olinfo.it/#/task/poldo/statement

Coin change – problema 1:
https://www.spoj.com/IIITM13A/problems/AASFPCB/

Coin change – problema 2:
https://www.codechef.com/problems/CTY2
or
https://www.hackerrank.com/challenges/coin-change/problem

Coin change – Problema 2
Suggerimento

$f(n, i)$ → numero di modi per ottenere valore complessivo *n* usando solo le monete di valore $c_o, c_1, \ldots, c_i$

Coin change – Problema 1
Harder challenge

Stampare le monete che vengono utilizzate in una soluzione ottima.

Ad esempio: N=10, M = 3, $C = \{1, 2, 4\}$
Stampare: 2 4 4

Infatti il minimo numero di monete da utilizzare per ottenere 10 è 3. Inoltre 2+4+4=10.
N.B: Potrebbero esserci più modi, basta stamparne uno.

Coin change – Problema 1
Harder challenge

Stampare le monete che vengono utilizzate in una soluzione ottima.

Suggerimento: osservate la funzione di ricorrenza $f(\cdot)$ che avete usato per risolvere il Problema 1. Vi fornisce informazioni su qual è la scelta della moneta migliore ad ogni passo?

Coin change – Problema 2
Harder challenge

Usa solo $O(N)$ di memoria. Non puoi dichiarare matrici $O(NM)$.

Suggerimento: modificare la soluzione bottom-up. Serve tenersi in memoria tutta la matrice, o basta tenerne una parte, man mano che vengono calcolate le soluzioni per gli stati?

Link esterni – altri problemi (DP)

Circa ordine di difficoltà

https://training.olinfo.it/#/task/trampolino/statement

https://training.olinfo.it/#/task/minato/statement

https://training.olinfo.it/#/task/discesa/statement

https://training.olinfo.it/#/task/scontri/statement

https://training.olinfo.it/#/task/cnn/statement

https://training.olinfo.it/#/task/ois_medaglie/statement

https://training.olinfo.it/#/task/ois_magnamagna/statement

https://training.olinfo.it/#/task/raisins/statement

https://training.olinfo.it/#/task/parole/statement (see KMP)

Spoiler per il problema Coin Change dopo questa slide

Coin change – Problema 1
Formula di ricorrenza

$$f(n) = \begin{cases} +\infty, & n < 0 \\ 0, & n = 0 \\ 1 + \min\limits_{c \in C} f(n-c), & n > 0 \end{cases}$$

Complexity

$O(N \cdot |C|)$

$$ans = f(N)$$

$if\ ans \geq +\infty$ $\rightarrow$ impossibile dare N con i valori in C

OSS: un valore accettabile come $+\infty$ potrebbe essere $N+1$ se i valori sono tutti interi

Coin change – Problema 2
Formula di ricorrenza

$$f(n,i) = \begin{cases} 0, & i < 0 \\ 1, & n = 0 \land i \geq 0 \\ f(n, c_{i-1}) & 0 < n < c \land i \geq 0 \\ f(n - c_i, c_i) + f(n, c_{i-1}) & else \end{cases}$$

Complexity

$O(N \cdot |C|)$

$$ans = f(N, M-1)$$