UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea

COMPOSIZIONE DI OPERATORI DI ORDINAMENTO CON CONTENITORI

COMPOSITION OF SORTING OPERATORS WITH CONTAINERS

ALESSIO SANTORO

Relatore: *Relatore*
Correlatore: *Correlatore*

Anno Accademico 2023-2024

"Inserire citazione"
— Inserire autore citazione

1

INTRODUZIONE

Lo scopo di questa introduzione é quello di introdurre i principali algoritmi di ordinamento che utilizzano sorting devices, in particolare stack-sort, queue-sort e bubble sort, e altri concetti necessari per l'analisi della loro composizione.
Durante l'eseguzione questi algoritmi possono salvare gli elementi in uno specifico contenitore (la diversa struttura adottata definise i diversi algoritmi) dalla quale poi vengono prelevati per essere aggiunti all'output.
Una sola iterazione non garantisce l'ordinamento della permutazione, dunque gli algoritmi devono essere iterati piú volte, ogni volta sul risultato della iterazione precedente. In ogni caso alla fine delle i-esima iterazione i maggiori i elementi avranno raggiunto la loro posizione finale, dunque sono necessari al massimo $n-1$ iterazioni per ordinare la permutazione.

L'oggetto di studio di questa tesi é la ricerca di condizioni che, per un operatore fissato, indicano che la permutazione é ordinabile con una sola iterazione, in particolare quando piú operatori vengono concatenati.

Essendo interessati al comportamento di una sola iterazione di questi algoritmi non si esaminerá tanto la consueta procedura di ordinamento, ma piuttoso un operatore, definito appositamente per ogni algoritmo, che descrive la singola iterazione. Ad esempio, prendendo l'algoritmo bubble-sort si fará riferimento all'operatore $B(\pi)$, dove π é una permutazione di interi, tale che n iterazioni del bubble-sort possano essere rappresentate da $B^n(\pi) = B(\dots B(\pi)\dots)$.

BUBBLE SORT

L'algoritmo di ordinamento bubble-sort prevede di scorrere gli elementi da ordinare dal primo al penultimo, ed ogni volta confrontare ogni

elemento con il suo successivo per scambiarli se non sono ordinati.
Il risultato si una singola iterazione di bubble-sort su una permutazione $\pi = \pi_1\pi_2 \ldots \pi_n$ é calcolato dall'operatore $B(\pi)$. Vale che sia $\pi = \pi_L n \pi_R$, dove n é il valore massimo, allora $B(\pi) = B(\pi_L)\pi_r n$.

Algorithm 1 operatore B - bubble sort, singola iterazione

for $i = 1$ **to** $n - 1$ **do**
  if $\pi_i > \pi_{i+1}$ **then**
    Swap π_i and π_{i+1}
  end if
end for

STACK SORT

L'operatore $S(\pi)$ il risultato ottenuto applicando stack sort su una permutazione π.
Il primo passo consiste nell'inserire π_1 nella pila. Poi lo si confronta con l'elemento π_2. Se $\pi_1 > \pi_2$ allora il secondo viene messo nella pila sopra π_1, altrimenti π_1 viene estratto dalla pila e inserito nell'output e π_2 viene inserito nella pila.
Gli stessi passi vengono eseguiti per tutti gli altri elementi presenti nell'input, se viene trovato un elemento nell'input maggiore dell'elemento in cima alla pila, la pila viene svuotata finché questa condizione non diviene falsa, poi l'elemento viene spinto nella pila.
Finiti gli elementi nell'input, se necessario, si svuota completamente la pila nell'output.
Sia $\pi = \pi_L n \pi_R$, con n valore massimo in π, vale che $S(\pi) = S(\pi_L)S(\pi_R)n$

Algorithm 2 operatore S - stack sort, singola iterazione

initialize an empty stack
for $i = 1$ **to** $n - 1$ **do**
  while unempty stack **and** π_i >peek **do**
    pop the stack in the output
  end while
  push(π_i)
end for
emtpy the stack in the output

QUEUE SORT

Per ogni elemento π_i della permutazione π in input se la coda é vuota o il suo ultimo elemento é minore di π_i, si accoda π_i, altrimenti si tolgono elementi dalla coda ponendoli nell'output fino a che l'elemento davanti alla coda non é maggiore di π_i, poi si aggiunge π_i all'output. Si svuota la coda nell'output.

Algorithm 3 operatore Q - queue sort, singola iterazione

```
initialize an empty queue
for i = 1 to n − 1 do
    if empty queue or last in queue < π_i then
        enqueue(π_i)
    else
        while first in queue < π_i do
            dequeue(π_i)
        end while
        add π_i to the output
    end if
end for
emtpy the queue in the output
```

OSSERVAZIONE *Bubble sort* é un caso particolare sia di *queue sort* che di *stack sort*.
Se infatti si fissa a 1 la dimensione della pila o della coda dei rispettivi operatori il comportamento che questi assumono é quello di una cella che, scorrendo l'input, contiene sempre il massimo valore trovato, mentre gli altri vengono messi nell'output.

CONTENITORI POP Un caso di studio interessante é quello in cui i contenitori di stack sort e queue sort vengano sostituiti dalla loro versione POP, cioé che quando viene eseguita un estrazione il contenitore viene svuotato completamente.

BYPASS Nel queue sort si puó osservare che a volte gli elementi vengono spostati direttamente dall'input all'output, senza passare dalla coda. Questa operazione é detta *bypass* e puó essere introdotta anche negli altri algoritmi.

CLASSI DI PATTERN DI PERMTUAZIONI

[1] Si scrive $\alpha \subseteq \beta$ per indicare che α é una sottosequenza di β, anche se non necessariamente una sottosequenza consecutiva.

Una permutazione δ é detta una sottopermutazione di una permutazione τ se τ ha una una sottosequenza che é un ordine isomorfico rispetto a δ, e si denota con $\delta \preceq \tau$. Ad esempio $312 \preceq 24153$ perché $413 \subseteq 24153$.

PATTERN 231 Se gli elementi a, b, c tali che $a < b < c$ compaiono nella permutazione in modo che b precede c e c precede a si dice che π contiene un pattern 231.
Quindi una permutazione π contiene un pattern 231 se $231 \preceq \pi$.

Questa é la notazione che verrá utilizzata per descivere i pattern esamianti.

CLASSI DI PATTERN La relazione di sottopermutazione é una relazione di ordine parziale che viene studiata con dei sottoinsiemi chiamati **pattern di classi**. Ogni classe di pattern D puó essere caratterizzata dall'insieme minimo M che evita:

$$D = Av(M) = \{\beta : \mu \npreceq \beta \forall \mu \in M\}$$

É noto in letteratura [2] che una permutazione puó essere ordinata da una sola passata di stack sort se e solo se non contiene pattern 231.
Allo stesso modo, sono note simile condizioni perché una permutazione possa essere ordinata da una sola passata degli altri operatoro descritti precedentemente.

Operatore	Permutazioni ordinabili con una sola passata
Satck sort	Av(231)
Queue sort	Av(321)
Bubble sort	Av(231,321)
Pop-stack sort	Av(231,312)
Pop-queue sort	Av(321,2413)
Pop-stack sort con bypass	Av(231,4213)

2

COMBINAZIONE DI OPERATORI

Si inizia mostrandp dei risultati ottenuti[1] riguardo alla combinazione degli operatori S e B.

BIBLIOGRAFIA

[1] Mathilde Bouvel, Lapo Cioni, and Luca Ferrari. Preimages under the bubblesort operator. *arXiv preprint arXiv:2204.12936*, 2022. (Cited on pages 6 and 7.)

[2] CONG HAN LIM. Brief introduction on stack sorting. (Cited on page 6.)