Corso di Algoritmi e Strutture Dati
Anno Accademico 2016–2017
Prof. Giovanni Pighizzini

Esercizi

1. Tecnica *divide et impera*

Esercizio 1.1
Calcolate il numero di spostamenti effettuati per il merge di due vettori di lunghezza n. Scrivete un'equazione di ricorrenza per il numero totale di spostamenti effettuati dall'algoritmo merge-sort. Risolvete poi l'equazione ottenuta.

Esercizio 1.2
Scrivete un algoritmo *divide et impera* che, quando $n > 1$, ordini un vettore di n elementi nel seguente modo:

- divide il vettore in tre parti ognuna di lunghezza circa $n/3$
- ordina ricorsivamente le parti ottenute
- effettua il merge della prima parte con la seconda e, successivamente, della parte risultante con la terza.

Scrivete un'equazione di ricorrenza per il numero di confronti. Risolvete l'equazione ottenuta.

Esercizio 1.3
Ripetete l'esercizio precedente supponendo di dividere il vettore in 4 parti. Per il merge si possono adottare diverse strategie, tra cui:

- effettuare il merge della prima parte con la seconda, del risultato con la terza e, infine, con la quarta,
- effettuare il merge della prima parte con la seconda, della terza con la quarta e, infine, effettuare il merge dei vettori risultanti.

Ci sono differenze tra le due strategie per quanto riguarda il numero di confronti?

Esercizio 1.4
Supponete di disporre di n vettori. Ciascun vettore è ordinato in modo crescente ed è costituito da n elementi. Utilizzando la tecnica *divide et impera* progettate un algoritmo per effettuare il merge di tali vettori. Stimate il numero totale di confronti, ricavando la corrispondente equazione di ricorrenza e risolvendola.

Ripetete poi l'esercizio supponendo di disporre di n vettori ordinati, contenenti al più m elementi. Stimate del numero totale di confronti in funzione di n e m.

Esercizio 1.5
Scrivete un algoritmo che riceva in ingresso un intero non negativo y e calcoli 99^y, nei seguenti modi:

- mediante prodotti iterati
- utilizzando la tecnica *divide et impera*.
  Suggerimento: potete ispirarvi allo schema ricorsivo utilizzato in un esempio della lezione 4 per il calcolo della potenza di una matrice.

In entrambi i casi calcolate il numero di prodotti effettuati. Date poi un limite superiore al tempo di calcolo (in funzione della lunghezza dell'input) utilizzando il criterio di costo logaritmico.

Esercizio 1.6
Ripetete l'esercizio precedente nel caso del calcolo di x^y, dove entrambi x e y sono ricevuti in input (potete considerare come stima della lunghezza dell'input la massima tra le lunghezze dei due numeri).

Esercizio 1.7
Scrivete un algoritmo che ricevendo in ingresso una matrice $n \times n$ A di numeri interi e un intero $m \geq 0$, calcoli la matrice A^m nei seguenti modi:

- mediante prodotti iterati,
- utilizzando la tecnica *divide et impera*.

In entrambi i casi calcolate in numero di prodotti di matrice effettuati in funzione di m. Stimate poi il numero totale di prodotti di interi effettuati (in funzione di n e di m) nei seguenti casi:

- se il prodotto di matrici viene effettuato applicando direttamente la definizione (prodotto righe per colonne),
- se il prodotto di matrici viene effettuato utilizzando l'algoritmo di Strassen.

Odd-even mergesort

Gli esercizi successivi illustrano una differente tecnica per il merge-sort, basata sullo schema *divide et impera*, proposta da Batcher nel 1968. Tale tecnica può essere utilizzata per progettare dei circuiti in grado di effettuare l'ordinamento. I circuiti sono basati su *comparatori*, dispositivi che ricevendo in ingresso due interi, li restituiscono in uscita in ordine crescente. Se cioè i due valori in ingresso sono x e y, sulla prima uscita sarà restituito $\min(x, y)$, mentre sulla seconda $\max(x, y)$ (vedi Figura 1.1).

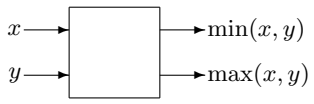

Figura 1.1: Un comparatore

Connettendo tra loro vari comparatori è possibile costruire delle reti che effettuino il merge e il sort. Ad esempio, il merge di due vettori di lunghezza 2 può essere effettuato con la rete di comparatori in Figura 1.2.

Esercizio 1.8
Il merge di due vettori ordinati $(a_1, \ldots, a_n)$ e $(b_1, \ldots, b_n)$ di lunghezza n, con n pari (indicato con Merge(n, n)), può essere effettuato utilizzando la rete rappresentata in Figura 1.3, secondo il seguente schema ricorsivo.

(i) Si effettua il merge delle due sequenze ordinate $(a_1, a_3, \ldots, a_{n-1})$ e $(b_1, b_3, \ldots, b_{n-1})$, formate dagli elementi nelle posizioni dispari dei vettori dati, sia $(d_1, \ldots, d_n)$ il vettore risultante.

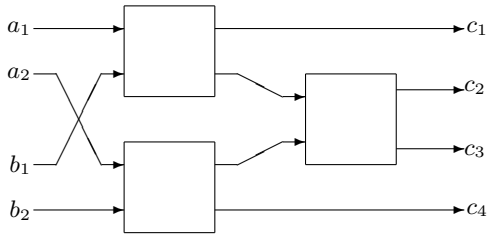

Figura 1.2: Rete di comparatori per Merge$(2, 2)$

(ii) Si effettua il merge delle due sequenze ordinate $(a_2, a_4, \ldots, a_n)$ e $(b_2, b_4, \ldots, b_n)$, formate dagli elementi nelle posizioni pari dei vettori dati, sia $(e_1, \ldots, e_n)$ il vettore risultante.

(iii) Il risultato del merge $(c_1, c_2, \ldots, c_{2n})$ può essere calcolato come segue:

- $c_1 = d_1$,
- $c_{2i} = \min(d_{i+1}, e_i)$, $c_{2i+1} = \max(d_{i+1}, e_i)$, per $i = 1, \ldots, n-1$,
- $c_{2n} = e_{2n}$

Scrivete un'equazione di ricorrenza e poi risolvetela, per il calcolo del numero di comparatori utilizzati per il merge di due vettori di lunghezza n (supponete che n sia potenza di 2).

Se ogni comparatore impiega tempo costante c per fornire il proprio risultato, quanto tempo serve per propagare il risultato dagli ingressi alle uscite della rete che effettua il merge? (Anche in questo caso scrivete e risolvete un'opportuna equazione di ricorrenza per ottenere la risposta.)

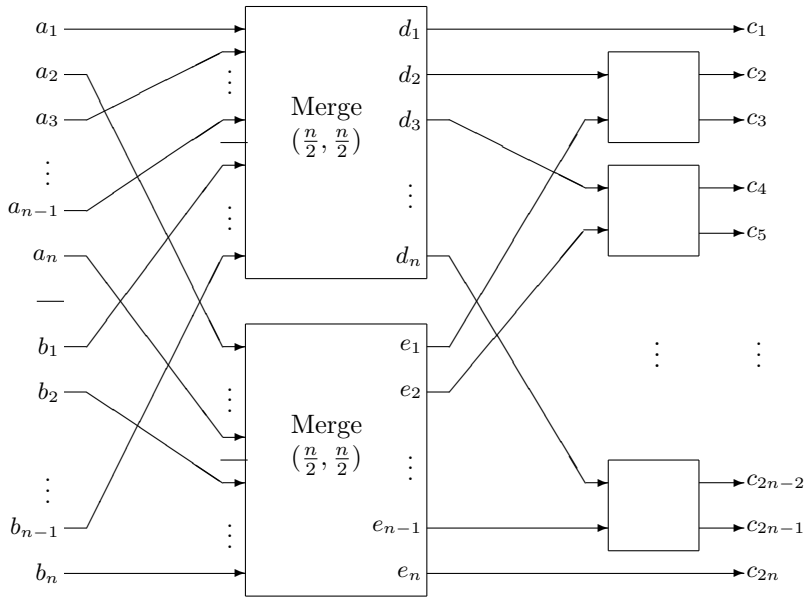

Figura 1.3: Schema di rete di comparatori per Merge(n, n)

Esercizio 1.9
Servendovi della rete di comparatori per effettuare il merge, potete costruire una rete di comparatori per il sort di due vettori di lunghezza n basandovi sullo stesso schema ricorsivo visto a lezione per l'algoritmo di merge-sort. Lo schema della rete è rappresentato in Figura 1.4.

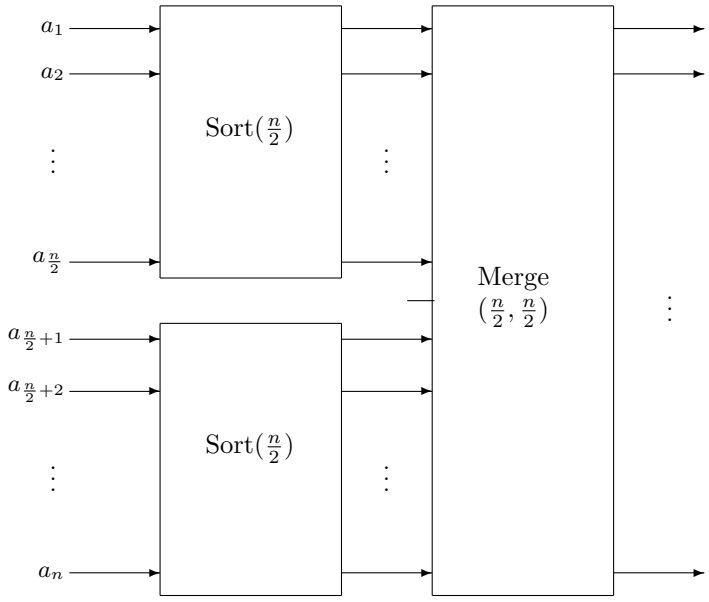

Figura 1.4: Schema di rete per il Sort(n)

Scrivete un'equazione di ricorrenza e poi risolvetela, per il calcolo del numero di comparatori utilizzati per il sort di due vettori di lunghezza n (supponete che n sia potenza di 2).

Se ogni comparatore impiega tempo costante c per fornire il proprio risultato, quanto tempo serve per propagare il risultato dagli ingressi alle uscite della rete che effettua il sort? (Anche in questo caso utilizzate un'opportuna equazione di ricorrenza per ottenere la risposta.)