ALGORITMI II (A.A. 2006-2007) Prova esame del 12 LUGLIO 2007

Esercizio 1 Un'impresa edile termina la costruzione di n nuovi appartamenti di diverse metrature e stipula un contratto con un'agenzia immobiliare che si impegna ad acquistarli tutti. Gli appartamenti hanno diversi costi, che indicheremo con $c_1, ..., c_n$, ma il loro valore di mercato decresce con il passare del tempo: in particolare, nel mese i l'appartamento x avrà un costo $\frac{c_x}{2^{i-1}}$.

L'agenzia immobiliare chiede di inserire nel contratto le seguenti clausole:

- si impegna ad acquistare tutti gli appartamenti entro n mesi;
- verrà acquistato almeno un appartamento al mese fino ad esaurimento degli appartamenti in vendita;
- l'appartamento/gli appartamenti acquistati in ogni mese potranno essere liberamente scelti tra quelli non ancora venduti.

L'impresa edile, per cautelarsi ed ottenere almeno un certo guadagno minimo m dalla vendita di ogni appartamento, fa aggiungere al contratto una ulteriore clausola: se un appartamento x non è acquistato dall'agenzia per vari mesi ed il suo prezzo diventa strettamente minore di m, l'impresa è svincolata dall'obbligo di cedere all'agenzia immobiliare quell'appartamento.

Assumendo che risulti $c_x \geq m$ per ogni appartamento x, l'agenzia immobiliare vuole garantirsi l'acquisto di tutti gli appartamenti minimizzando la spesa.

Ad esempio, siano $n = 5$, $m = 11$, $c_1 = 100$, $c_2 = 20$, $c_3 = 80$ e $c_4 = 20$.

- L'acquisto di tutti gli immobili nel primo mese soddisfa il requisito dell'impresa edile (il guadagno su ogni appartamento è ≥ 11) e comporta per l'agenzia una spesa $100 + 20 + 80 + 20 = 220$.
- L'acquisto degli immobili 1 e 4 nel primo mese, dell'immobile 2 nel secondo mese e dell'immobile 3 nel terzo mese non non garantisce la possibilità di acquistare tutti gli appartamenti: il guadagno sull'immobile 2 sarebbe infatti $\frac{20}{2} < 11$, e quindi l'agenzia non avrebbe la certezza di ottenere quell'immobile.
- L'acquisto degli immobili 1, 2 e 4 nel primo mese e dell'immobile 3 nel terzo mese non è possibile, perché nel secondo mese non viene acquistato alcun appartamento ma ci sono ancora appartamenti non venduti.
- L'acquisto degli immobili 1, 2 e 4 nel primo mese e dell'immobile 3 nel secondo mese è invece legale e comporta una spesa pari a $100 + 20 + 20 + \frac{80}{2} = 180$.
- La spesa minima in questo esempio è ottenuta acquistando gli immobili 2 e 4 nel primo mese, l'immobile 3 nel secondo mese e l'immobile 1 nel terzo mese, ed è pari a $20+20+80/2+100/4 = 105$.

1. *(max 10 punti)* Descrivere lo pseudocodice di un algoritmo che dia in output per ogni appartamento il mese in cui l'agenzia immobiliare può acquistarlo in modo da minimizzare la spesa e garantirsi comunque l'acquisto di tutti gli appartamenti. L'algoritmo deve avere un tempo di calcolo $O(n \cdot \log n)$.
2. *(max 10 punti)* Provare la correttezza dell'algoritmo proposto.

Soluzione Esercizio 1 L'ideale sarebbe acquistare ciascun appartamento nell'ultimo mese utile prima che il suo costo risulti inferiore a m. In questo modo l'agenzia acquisterebbe tutti gli appartamenti, ciascuno al costo più basso possibile. Tuttavia così facendo potrebbe accadere che in alcuni mesi non viene acquistato nulla violando la clausola del contratto che prevede che almeno un'appartamento al mese sia acquistato. Modifichiamo quindi la strategia in modo che, finché ci sono appartamenti, in ogni mese vengono acquistati tutti quelli in "scadenza" in quel mese (vale a dire quelli che se non acquistati nel mese successivo finiscono con l'avere un costo inferiore ad m). Inoltre se nessun appartamento è in scadenza in un dato mese, comunque ne viene acquistato uno (quello che costa meno). E' possibile risolvere il problema in tempo $O(n \log n)$ ordinando preliminarmente gli appartamenti per costo iniziale non decrescente ed utilizzando il seguente algoritmo greedy (che assume $c_1 \leq c_2 \leq ... \leq c_n$):

INPUT: n, m e costo iniziale c_i di ogni appartamento, con $c_1 < c_2 < ... < c_n$.
OUTPUT: la spesa dell'agenzia e l'array SOL (di dimensione n) tale che $SOL[i]$ indica il mese in cui l'agenzia acquista l'appartamento i.

```
i ← 1
mese ← 1
fattore ← 1
WHILE i ≤ n DO
    SOL[i] ← mese
    i ← i + 1
    WHILE c_i / (2 · fattore) < m DO
        SOL[i] ← mese
        i ← i + 1
    ENDWHILE
    mese ← mese + 1
    fattore ← 2 · fattore
ENDWHILE
RETURN SOL
```

L'algoritmo considera gli appartamenti per costi non decrescenti e mantiene un contatore $mese$ che indica il mese corrente. Osserviamo che il costo di un appartamento i nel mese $mese$ è $c_i/fattore$. Nel considerare l'appartamento i, se assegnando i al mese successivo (il $(mese+1)$-esimo) il vincolo imposto dall'impresa edile viene violato, allora i viene assegnato al mese corrente. Altrimenti il contatore del mese viene incrementato e quindi i viene assegnato al mese successivo.

E' facile vedere che il tempo di esecuzione dell'algoritmo è $\Theta(n)$, tranne per l'ordinamento dei costi iniziali che richiede tempo $\Theta(n \log n)$.

Discutiamo ora la correttezza dell'algoritmo proposto. Seguendo lo schema di prova per gli algoritmi greedy proposto in classe possiamo dimostrare (per induzione) che, in ogni istante i, esiste una soluzione ottima SOL^* che include la soluzione parziale SOL_i trovata nell'algoritmo. La prova è lasciata per esercizio.

Esercizio 2 Sia S un insieme di n valori. Una permutazione $v_1, v_2, ...v_n$ degli elementi di S si dice *ordinata a zig-zag* se v_1 è il minimo dell'insieme S, v_2 il massimo, v_3 il secondo minimo, v_4 il secondo massimo, e via dicendo. Ad esempio, se $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, la sequenza 1 9 2 8 3 7 4 6 5 è ordinata a zig-zag. Progettare un algoritmo basato sull tecnica del *divide et impera* che, dato un array disordinato di n elementi, ne produca un ordinamento a zig-zag in tempo $\Theta(n \log n)$.

Soluzione Esercizio 2 Basta modificare opportunamente l'algoritmo mergesort, ed in particolare la funzione merge. L'array A in input viene diviso a metà, e le due metà A_1 e A_2 sono ricorsivamente ordinate a zig-zag. Dai due ordinamenti a zig-zag è facile produrre un ordinamento a zig-zag dell'intera sequenza in tempo $\Theta(n)$. Ad esempio, si possono eseguire i seguenti passi:

1. produrre da A_1 un array B_1 ordinato in modo classico;
2. produrre da A_2 un array B_2 ordinato in modo classico;
3. produrre da B_1 e B_2 un array B ordinato in modo classico, contenente tutti gli elementi. Per far ciò basta usare il merge classico adoperato dall'algoritmo mergesort;
4. produrre da B l'array A ordinato a zig-zag.

Con un approccio leggermente più sofisticato è anche possibile produrre A in modo diretto da A_1 ed A_2, senza passare per gli array ordinati ausiliari. Il tempo di esecuzione per la fuzione è comunque lineare.

La ricorrenza che descrive il tempo di esecuzione dell'intero algoritmo di ordinamento a zig-zag è identica a quella del mergesort, con soluzione $\Theta(n \log n)$.