

Esame scritto ASD 26 luglio 2022

Tempo totale 45 minuti; punteggio massimo 6.9; sufficienza 4 punti

- Se hai una certificazione DSA rispondi alle domande 1 e 2
- Se non hai una certificazione DSA rispondi a tutte le domande

Domanda #1

In uno Heap Binario di tipo "min" la radice ha chiave minima e ogni nodo discendente di un nodo N deve avere etichetta maggiore di quella di N: le operazioni sono analoghe a quelle di uno Heap Binario di tipo max come visto a lezione, invertendo "<" e ">".

[Preparazione dei dati, senza voto; la preparazione dei dati non viene svolta come indicato l'esercizio non verrà corretto] Considerate le prime 4 cifre della vostra matricola e scrivete prima di ciascuna di esse 8, 6, 4, 2 rispettivamente. Ad esempio, se la vostra matricola è 1234567 le prime quattro cifre sono 1, 2, 3, 4: dovete porre prima di ciascuna di esse 8, 6, 4, 2, ottenendo la sequenza 8 1 6 2 4 3 2 4

Considerate la sequenza così ottenuta come coppie di cifre, e ciascuna cifra come un numero intero: nell'esempio si ottengono 8 1 6 2 4 3 2 4, quindi i numeri interi 81, 62, 43, 24.

[1/2 del punteggio] Si consideri uno heap binario A di tipo min inizialmente vuoto. Si disegni come viene modificato A dopo l'inserimento del primo numero intero ottenuto al passo precedente (81 nell'esempio). Poi si disegni come lo heap ottenuto viene modificato con l'inserimento del secondo (62 nell'esempio). Poi si disegni lo heap dove viene inserito anche il terzo numero e infine si disegni lo heap in cui viene inserito anche il quarto numero, senza fornire alcuna spiegazione dei passaggi.

[1/2 del punteggio] Si spieghino dettagliatamente, mediante testo tecnicamente corretto in ogni affermazione e disegni chiari (dovete fornire sia testo che disegni), i passaggi della chiamata di `teMin(A)`; effettuata sullo heap A ottenuto nel punto precedente.

Domanda #2

Si consideri una tabella di hash con liste di collisione che implementa un dizionario D in cui le chiavi sono stringhe di 4 cifre decimali che indichiamo con c1, c2, c3, c4 e i valori sono stringhe. La tabella di hash ha 7 bucket indicizzati da 0 a 6 e la funzione di hash è h: $(c1 + c2 \cdot c3 + c4) \bmod 7$.

[Preparazione dei dati, senza voto] Scrivete sul foglio della risposta le cifre 0 0 7 0 5, poi la vostra data di nascita nel formato ANNO (4 cifre) MESE (2 cifre) GIORNO (2 cifre), poi la vostra matricola universitaria (7 cifre), poi la vostra matricola universitaria al contrario (7 cifre), e infine le cifre 0 1 7 0

Ad esempio, se la vostra matricola è 1234567 e siete nati il primo gennaio 2000, dovete scrivere sul foglio

0 0 7 0 5 2 0 0 0 0 1 0 1 1 2 3 4 5 6 7 7 6 5 4 3 2 1 0 1 7 0
0 0 7 0 5 data nascita matricola matricola al contrario 0 1 7 0

[Preparazione dei dati, senza voto MA LA PRESENZA DI DUE O PIÙ ERRORI NEL CALCOLO DELLA FUNZIONE DI HASH COMPORTA L'ASSEGNAZIONE DI ZERO PUNTI ALL'INTERO ESERCIZIO] Disegnate sul foglio della risposta uno schema come quello mostrato in seguito e calcolate la funzione di hash delle chiavi che otteneute:

- nella prima riga dovreste scrivere il numero che avete generato al Passo 1, diviso in gruppi di quattro cifre: questi gruppi rappresenteranno le chiavi

- nella seconda riga, i valori associati alle chiavi, ovvero le stringhe "ma", "co", "gra", "pe", "le", "re", "ca", "giu", "ca", "giu"
- nella terza riga, il valore di hash associato a ogni chiave, che dovrebbe calcolare usando la formula h: $(c1 + c2 \cdot c3 + c4) \bmod 7$ (ricordiamo che * ha precedenza su +)

0 0 7 0	5 0 2 0	0 0 0 1	0 1 1 2	3 4 5 6	7 7 6 5	4 3 2 1	0 1 7 0
ma	co	gra	pe	le	re	ca	giu
valore calcolato di h(0070)	valore calcolato di h(5020)	valore calcolato di h(0001)

[1/3 del punteggio] Il dizionario D implementato dalla tabella di hash è inizialmente vuoto. Disegnate la tabella di hash che si ottiene dagli inserimenti delle stringhe "ma", "co", "gra", "pe", "le", "re", "ca", "giu" associate alle chiavi su quattro cifre così come le avete calcolate al Passo 2, nell'ordine in cui le avete scritte da sinistra verso destra. Nel caso ci fossero delle chiavi duplicate, ignorate l'inserimento e scrivete esplicitamente sul foglio "questa coppia chiave-valore è stata ignorata perché la chiave è duplicata". Continuando l'esempio, si dovrebbe disegnare la tabella di hash risultante da:

```
insert(D, "00070", "ma").  
insert(D, "5020", "co").  
insert(D, "0001", "gra").  
....
```

[1/3 del punteggio] quali sono le due principali "buone proprietà" che una funzione di hash deve avere? Rispetto alle chiavi che avete ottenuto al Passo 2, la funzione di hash h applicata alle chiavi ottenute al Passo 2 ha queste due "buone proprietà"? Motivate esaurientemente la risposta. Risposte generiche e non relative alla funzione h applicata alle chiavi così come ottenute al Passo 2 comportano l'assegnazione di 0 punti.

[1/3 del punteggio] sotto l'ipotesi che la funzione di hash abbia tutte le "buone proprietà" che dovrebbe avere, qual è la complessità dell'operazione di cancellazione di un elemento data la sua chiave nel caso peggiore? Non basta copiare la risposta dalle slide del corso o da altro materiale didattico: è necessario spiegare cosa rappresentano i parametri che caratterizzano la risposta e motivarla esaurientemente. La mancanza di motivazione chiara ed esauriente comporta l'assegnazione di 0 punti a questa domanda.

Domanda #3 (SOLO PER STUDENTI SENZA CERTIFICAZIONE DSA)

[1/2 punteggio] Si scriva quanto vale la complessità di `quicksort` nel caso peggiore e si illustri dettagliatamente quando si ricade nel caso peggiore (la risposta "quando le due sottosequenze sono sbilanciate" non è sufficiente: bisogna spiegare sotto quali condizioni accade che siano sbilanciate) e come si calcola la complessità di quicksort nel caso peggiore, senza fare uso del Teorema Master che non è parte del programma di ASD. Rispetto alla sequenza riportata sotto, si illustriano i primi due passaggi dell'esecuzione di quicksort nel caso peggiore.

30 25 20 15 50 45 40 35

[1/2 punteggio] Si simuli una esecuzione della chiamata di `mergesort` sulla sequenza riportata sotto. Questa situazione coincide con il caso migliore di mergesort, con il caso peggiore, o con il caso medio? Qual è la complessità di mergesort in questa situazione?

50 45 40 35 30 25 20 15

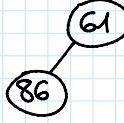
Domanda 1.1

$$6145 \dots \rightarrow 86, 61, 44, 25$$

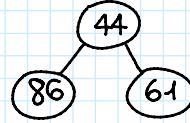
1) inser. primo elem.



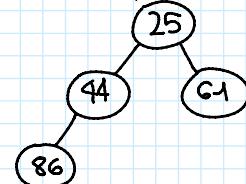
2) inser. secondo elem.



3) inser. terzo elem.

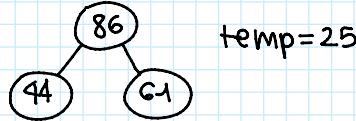


4) inser. quarto elem.

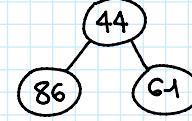


Domanda 1.2

1) Salvo l'elemento nel nodo della radice, poi scambio il nodo della radice con l'ultimo nodo disponibile



2) entrambi i figli sono più piccoli del padre (86), quindi scambio quest'ultimo con il più piccolo, poi ritorno temp.



Domanda 2.1

007050 20040701 6145273 3725416 0170

0 0 7 0	5 0 2 0	0 4 0 7	0 1 6 1	4 5 2 7	3 3 7 2	5 4 1 6	0 1 7 0
ma	co	gra	pe	le	re	ca	giu
0	5	0	0	0	5	1	0

$$(c1 + c2 \cdot c3 + c4) \bmod 7$$

$$0+0\cdot7+0=0+0+0=\frac{0}{0}$$

$$4+5\cdot2+7=4+10+7=\frac{21}{0}$$

$$5+0\cdot7+0=5+0+0=5$$

$$3+3\cdot7+9=3+21+9=26$$

$$(c_1 + c_2 \cdot c_3 + c_4) \bmod 7$$

$$0+0 \cdot 7+0=0+0+0=\frac{0}{0} \quad 7$$

$$5+0 \cdot 2+0=5+0+0=\frac{5}{5} \quad 7$$

$$0+4 \cdot 0+7=0+0+7=\frac{7}{0} \quad 7$$

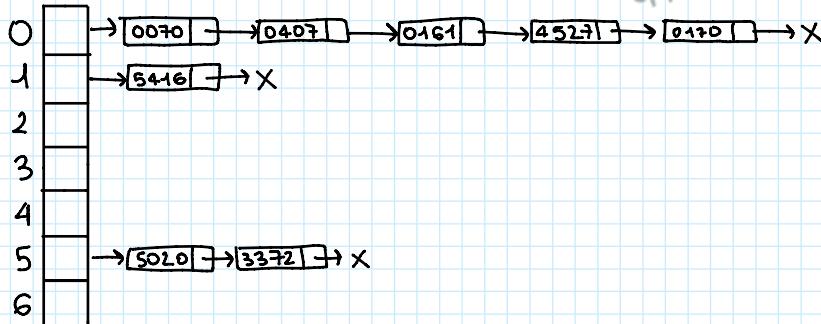
$$0+1 \cdot 6+1=0+6+1=\frac{7}{0} \quad 7$$

$$4+5 \cdot 2+7=4+10+7=\frac{21}{0} \quad 7$$

$$3+3 \cdot 7+2=3+21+2=\frac{26}{5} \quad 7$$

$$5+4 \cdot 1+6=5+4+6=\frac{15}{7} \quad 7$$

$$0+1 \cdot 7+0=0+7+0=\frac{7}{0} \quad 7$$



Domanda 2.2

Le due principali buone proprietà che una tabella di hash dovrebbe avere sono quella di essere calcolabile in tempo costante e l'essere uniforme. La prima proprietà è soddisfatta, in quanto la funzione di hash richiede operazioni aritmetiche semplici, il cui tempo di esecuzione non dipende dall'input. La seconda proprietà non viene rispettata, in quanto alcune chiavi generano più frequentemente la stessa chiave: infatti il bucket 0 contiene maggior parte delle chiavi, mentre i bucket 2, 3, 4 e 6 rimangono vuoti.

Domanda 2.3

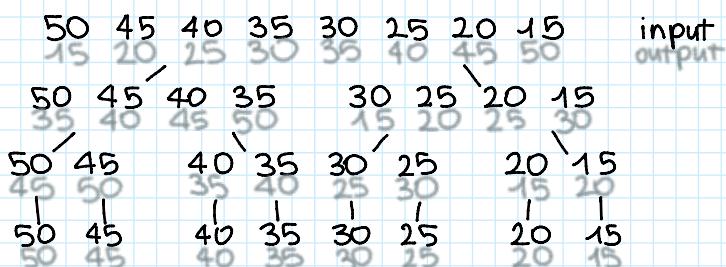
Nel caso peggiore, la complessità è $O(n/m)$. Questo perché se la funzione di hash ha tutte le "buone proprietà", allora ogni bucket ha n/m elementi. Quindi se la ricerca avviene in $O(n/m)$ e la cancellazione avviene in $O(1)$, la complessità della delete è $O(n/m \cdot 1) = O(n/m)$

Domanda 3.1

Nel caso peggiore, quicksort ha complessità $O(n^2)$ e si ricade in questo caso quando si sceglie come pivot l'elemento minore o maggiore della sequenza. Per calcolare la complessità si sommano le operazioni fatte ad ogni chiamata ricorsiva. La "prima volta" si eseguono n operazioni, la seconda volta $n-1$ operazioni e così via fino all'ultimo livello in cui si esegue 1 operazione. Questo calcolo si sviluppa nella sommatoria per "i" che va da 1 a n . Quindi $n+(n-1)+(n-2)+\dots+1 = n^2$.

$$\begin{array}{ccccccccc}
 30 & 25 & 20 & 15 & 50 & 45 & 40 & 35 \\
 30 & 25 & 20 & 15 & 45 & 40 & 35 & 50 \\
 30 & 25 & 20 & 15 & 40 & 35 & 45 & 50
 \end{array}$$

Domanda 3.2



Ci troviamo nel caso peggiore, in quanto l'array è in ordine decrescente. Tuttavia, secondo un algoritmo ordinativo il momento effettivo tutto lo chiamato ricorsivo

50 45 40 35 30 25 20 15
Ci troviamo nel caso peggiore, in quanto l'array è in ordine decrescente. Tuttavia, essendo un algoritmo adattivo, il mergesort effettua tutte le chiamate ricorsive necessarie, e tutti i confronti, anche nel caso in cui l'array sia già ordinato. La complessità è quindi $O(n \log n)$.