

Esame ASD 6 settembre 2021 - 3 domande (gli studenti/studentesse con certificazione DSA o che sostengono l'esame in modalità integrata devono fare solo le prime due e devono scrivere sul foglio, accanto al nome, "DSA" oppure "INTEGRATA"); tempo totale 45 minuti; punteggio massimo 6.9; sufficienza 4 punti

Domanda #1 (per tutti)

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: $(c1+c2+c3+c4) \bmod 7$.

Passo 1 (senza voto): scrivete sul foglio della risposta le cifre 1 2 1 2 0 0, 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 1 2 0 1

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

1	2	1	2	0	0	0	0	1	0	1	2	3	4	5	6	7	6	5	4	3	2	1			
1	2	1	2	0	0	0	0	1	0	1	2	0	1	2	3	4	5	6	7	6	5	4	3	2	1

Passo 2 (senza voto, MA LA PRESENZA DI DUE O PIU' 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 sotto e calcolate la funzione di hash delle chiavi che otteneate:

- nella prima riga dovreste scrivere il numero che avete generato al Passo 1, diviso in gruppi di quattro cifre: questi gruppi rappresentano le chiavi
- nella seconda riga, i valori associati alle chiavi, ovvero le stringhe "maritime", "cozie", "gracie", "pennine", "lepontine", "retiche", "carniche", "giulie"
- nella terza riga, il valore di hash associato a ogni chiave, che dovreste calcolare usando la formula h: $(c1+c2+c3+c4) \bmod 7$

1	2	1	2	0	0	0	1	0	1	2	3	4	5	6	7	6	5	4	3	2	1
maritime	cozie	gracie	pennine	lepontine	retiche	carniche	giulie														
val. di h(1212)	val. di h(0020)	val. di h(0001)														

Passo 3 (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 "maritime", "cozie", "gracie", "pennine", "lepontine", "retiche", "carniche", "giulie" 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. Ricordate che in un dizionario non si possono ammettere chiavi duplicate: nel caso ce ne fossero ignorate l'inserimento del duplice 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, "1212", "maritime").
insert(D, "0020", "cozie").
insert(D, "0001", "gracie").
....
```

Passo 4 (1/3 del punteggio): quali sono le due principali "buone proprietà" che una tabella 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.**

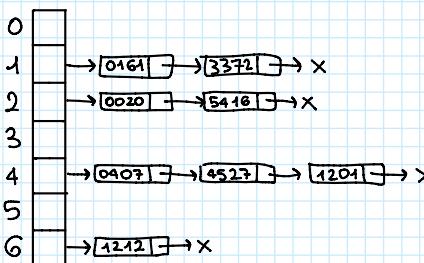
Domanda 1.1

121200 20040701 6145273 3725416 1201

h: $(c1+c2+c3+c4) \bmod 7$

1	2	1	2	0	0	0	1	0	1	2	3	4	5	6	7	6	5	4	3	2	1
maritime	cozie	gracie	pennine	lepontine	retiche	carniche	giulie														
6	2	4	1	4	1	2	4														

6 mod 7 2 mod 7 11 mod 7 8 mod 7 18 mod 7 15 mod 7 16 mod 7 4 mod 7



Domanda 1.2

Le due principali buone proprietà sono l'essere calcolabile in tempo costante e l'uniformità. La prima proprietà è soddisfatta in quanto la funzione di hash viene calcolata sempre su chiavi formate da 4 cifre; inoltre, le operazioni aritmetiche del calcolo sono semplici, quindi calcolabili in tempo costante (somma e modulo). La seconda proprietà invece non è soddisfatta: alcuni bucket sono vuoti (0, 3, 5), e altri più pieni di altri (4); questo significa che alcune chiavi generano più frequentemente lo stesso valore, quindi l'uniformità non è possibile.

Domanda 1.3

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 3.1.1

La complessità di mergesort è $O(n \log n)$ sia nel caso migliore che nel caso peggiore, infatti, non essendo un algoritmo

Passo 5 (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

[1/2 punteggio] Si scriva quanto vale la complessità di mergesort nel caso migliore e nel caso peggiore e si illustri dettagliatamente quando si ricade nel caso peggiore, quando si ricade nel caso migliore, e come si calcola la complessità nei due casi.

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

5 6 8 7 9 10 3 4 2 1

Domanda #3 (se siete in possesso di una certificazione non dovete rispondere a questa domanda)

[1/2 punteggio] Si scriva quanto vale la complessità di quicksort nel caso peggiore e si illustri dettagliatamente quando si ricade nel caso peggiore e come si calcola la complessità di quicksort nel caso peggiore.

[1/2 punteggio] Si simuli l'esecuzione della chiamata di quicksort sulla sequenza riportata sotto, con il pivot scelto sempre, "per magia", come elemento mediano della sottosequenza da riordinare. Questa situazione coincide con il caso migliore di quicksort, con il caso peggiore, o con il caso medio? Qual è la complessità di quicksort in questa specifica situazione?

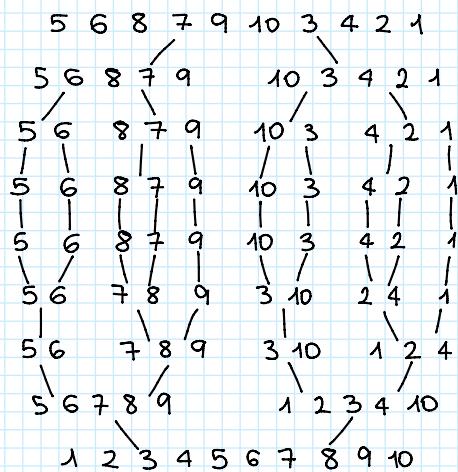
5 6 8 7 9 10 3 4 2 1

In $O(1)$, la complessità della delete è $O(n/m \cdot 1) = O(n/m)$.

Domanda 3.1.1

La complessità di mergesort è $O(n \log n)$ sia nel caso migliore che nel caso peggiore, infatti, non essendo un algoritmo adattivo, effettua tutte le chiamate e i confronti necessari in qualsiasi caso. Il caso peggiore avviene quando l'array è in ordine decrescente o in disordine, mentre il caso migliore è quando è già ordinato. La complessità deriva dal prodotto tra il numero dei livelli e il costo di ciascun problema. Ad ogni livello ci sono 2^e problemi ognuno lungo $n/2^e$ e quindi risolvibili in $O(n/2^e)$. Il costo di ogni livello sarà quindi $2^e \cdot n/2^e = O(n)$. Se il numero di livelli è $\log n$, la complessità sarà $O(n) \cdot \log n = O(n \log n)$.

Domanda 3.1.2

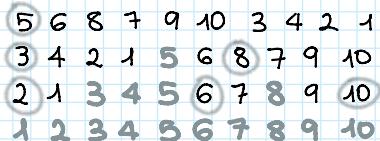


Questa situazione coincide con il caso peggiore, in quanto l'array non è ordinato. La complessità è $O(n \log n)$.

Domanda 3.2.1

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 $n + (n-1) + (n-2) + \dots + 1 = n^2$.

Domanda 3.2.2



Questa situazione coincide con il caso migliore del quicksort, quindi la complessità è $O(n \log n)$.