

Progetto “Filtro di Bloom”

Simulazione del 4 Giugno 2020

Descrizione del progetto

Lo scopo del progetto è l'implementazione di un *filtro di Bloom*, una struttura dati che consente di stabilire (con una bassa probabilità di errore) se un certo elemento è già stato visto o no. In altre parole, un filtro di Bloom può essere utilizzato per implementare un insieme di elementi in cui le operazioni consentite sono l'inserimento e la verifica se l'elemento esiste già nell'insieme e questa ultima operazione può ritornare un risultato errato ma con una bassa probabilità.

L'idea del filtro di Bloom nasce dalla necessità di avere una rappresentazione per degli insiemi che sia più compatta della memorizzazione di tutti gli elementi contenuti. Questa rappresentazione non è però perfetta e può dare la risposta sbagliata, potremmo quindi dire che un elemento è presente nell'insieme quando in realtà non lo è. L'errore opposto (dire che un elemento è assente quando in realtà è presente) risulta però impossibile. Il filtro di Bloom è utile quando (rari) errori di questo tipo sono comunque accettabili: vengono utilizzati, per esempio, all'interno di Google Chrome, Bing, Medium, etc.

Un filtro di Bloom consiste di un array a di m elementi (che andranno ad assumere solo valori binari) e k funzioni di hash distinte h_1, \dots, h_k che abbiano range $0, \dots, m-1$, ovvero tutte le possibili posizioni nell'array a .

Inizialmente l'array a contiene il valore 0 in tutte le posizioni. Quando un elemento x viene inserito la procedura è la seguente:

- Vengono calcolate $h_1(x), h_2(x), \dots, h_k(x)$
- I valori di a nelle posizioni $h_1(x), h_2(x), \dots, h_k(x)$ sono impostati a 1

Per verificare se un elemento x è già stato inserito in precedenza la procedura è la seguente:

- Vengono calcolare $h_1(x), h_2(x), \dots, h_k(x)$
- Se i valori di a nelle posizioni $h_1(x), h_2(x), \dots, h_k(x)$ sono tutti 1 allora viene ritornato **True**
- Altrimenti viene ritornato **False**

Risulta quindi che se viene ritornato **False** sicuramente l'elemento x non era mai stato inserito in precedenza (a nelle posizioni $h_1(x), h_2(x), \dots, h_k(x)$ avrebbe avuto valore 1). È invece possibile sbagliare nel caso si ritorni **True**, dato che i valori nelle posizioni $h_1(x), h_2(x), \dots, h_k(x)$ potrebbero essere stati impostati a 1 dall'inserimento di altri elementi.

La probabilità di errore dipende quindi dal numero di funzioni di hash k e dalla dimensione dell'array m , permettendo quindi avere un compromesso tra la probabilità di avere un errore e lo spazio utilizzato dal filtro di Bloom. In particolare la probabilità di un errore dato il numero di elementi inseriti n , la

dimensione dell'array m e il numero di funzioni di hash k è approssimativamente $(1 - e^{kn/m})^k$.

Funzioni di hash

Mentre nel caso di filtri di Bloom per utilizzi reali la scelta delle funzioni di hash è importante, per questo progetto si utilizzeranno solo due funzioni di hashing $h_1(x) = x \bmod m$ e $h_2(x) = 1 + (x \bmod (m - 1))$ e come famiglia di funzioni di hashing si utilizzeranno $g_i(x) = (h_1(x) + ih_2(x)) \bmod m$ per $i = 0, \dots, k - 1$. Si noti che la famiglia di funzioni $g_i(x)$ può essere vista come una unica funzione di hashing con due argomenti $g(x, i) = g_i(x)$, come abbiamo visto nel caso del double hashing.

Codice

Il codice deve contenere l'implementazione di una classe `FiltroDiBloom` con i seguenti metodi:

- un costruttore che ha come argomenti la dimensione `m` del filtro ed il numero di funzioni di hash da utilizzare `h`;
- un metodo `inserisci` che inserisce l'elemento `x` nell'insieme. Si ricorda che questo non consiste nel salvare l'elemento ma nell'impostare alcuni dei valori nell'array di lunghezza `m`;
- un metodo `esiste` che ritorna un valore Booleano che serve a determinare se `x` sia già stato inserito in precedenza o no. Essendo il filtro di Bloom una struttura probabilistica, accettiamo che possa essere restituito `True` in modo erroneo. Per come i filtri di Bloom sono costruiti, il valore `False` viene ritornato solo quando l'elemento `x` non è mai stato inserito in precedenza.

Uno scheletro del programma è il seguente:

```
class FiltroDiBloom:

    # m è la dimensione della tabella che utilizziamo
    # k è il numero di funzioni di hash
    def __init__(self, m, k):
        # codice

    # x è l'elemento da inserire nell'insieme.
    # Il metodo non deve ritornare nulla
    def inserisci(self, x):
        # codice

    # x è l'elemento che vogliamo verificare se sia già presente
    # il metodo deve ritornare true se x è presente (potenzialmente
```

```
# questo risultato può essere errato) e false altrimenti  
def esiste(self, x):  
    # codice
```

Il codice può anche contenere funzioni e metodi aggiuntivi a seconda di come viene svolta l'implementazione.

Indicazioni

Il progetto deve essere svolto **individualmente**. La consegna dovrà avvenire entro le ore 23:59 del giorno xx/yy/yyyy secondo le seguenti modalità:

- Invio di una email a `lmanzoni@units.it` dal vostro account email istituzionale con oggetto *[informatica] consegna progetto appello del xx/yy/yyyy*.
- L'email deve avere come allegato il progetto in un singolo file in codice sorgente Python versione 3, dal nome `Nome_Cognome_matricola.py`, quindi, per esempio Mario Rossi di matricola 12345 consegnerà un file dal nome `Mario_Rossi_12345.py`.
- Il file deve contenere sotto forma di commento le seguenti linee indicanti nome, cognome e numero di matricola:

```
# Nome: Mario  
# Cognome: Rossi  
# Matricola: 12345
```