Giorno 5

1 Pillole di probabilità

1.1 Probabilità

Sia S un insieme detto *spazio dei campioni*, i cui elementi, detti eventi elementari, sono i possibili risultati di un esperimento.

1.1.1 Assiomi di probabilità, distribuzioni di probabilità discrete

Gli *eventi* sono elementi di $\mathcal{P}(S)$.
Sia $\mathrm{pr}\{\cdot\} \subseteq \mathcal{P}(S) \times \mathbb{R}$ una relazione che associa ad ogni evento di S un numero reale, in modo che siano soddisfatti gli *assiomi di probabilità*:

1. $\Pr\{S\} = \sum_{s \in A} \Pr\{s\} = 1$
2. $\forall s \in \text{eventi}(S) : \Pr\{s\} \geq 0$
3. $\Pr\{A \cup B\} = \Pr\{A\} + \Pr\{B\} - \Pr\{A \cap B\}$

Due eventi tali che $\Pr\{A \cap B\} = \emptyset$ si dicono *mutualmente esclusivi*;
Gli eventi elementari sono mutualmente esclusivi.

Se S è finito e ogni elemento di S ha probabilità $1/|S|$, allora si ha una *distribuzione di probabilità uniforme.*

1.1.2 Probabilità condizionata e indipendenza

La probabilità condizionata di un evento A, sapendo che si è verificato un evento B, è definita come:

$$\Pr\{A|B\} = \frac{\Pr\{A \cap B\}}{\Pr\{B\}}$$

Due eventi sono indipendenti se $\Pr\{A \cap B\} = \Pr\{A\}\Pr\{B\}$

1.2 Variabili aleatorie discrete

Una variabile aleatoria X è una funzione $X : S \to \mathbb{R}$, dove S può essere finito o infinito numerabile.
Definiamo l'evento $X = x$ come $s \in S : X(s) = x$. Si ha:

$$\Pr\{X = x\} = \sum_{s \in S : X(s) = x} \Pr\{s\}$$

La funzione $f(x) = \Pr\{X = x\}$ è detta *funzione densità di probabilità.*

1.2.1 Valore atteso

Il valore atteso (o medio) di una variabile aleatoria X è $E[X] = \sum_{x} x\Pr\{X = x\}$.

2 Dizionari

Un dizionario è una struttura dati su cui sono definite le operazioni di inserimento, ricerca e cancellazione. Un dizionario può essere implementato su varie SD, in particolare analizziamo le seguenti:

1. Array non ordinato:

Ricerca	**Inserimento**	**Cancellazione**
Sequenziale - $O(n)$	$O(1)$	Non lasciare buchi - $O(n)$

2. Array ordinato:

Ricerca	**Inserimento**	**Cancellazione**
Binaria - $O(\log n)$	Ordinato - $O(n)$	Non lasciare buchi - $O(n)$

3. Lista non ordinata:

Ricerca	**Inserimento**	**Cancellazione**
$O(n)$	In testa - $O(1)$	$\begin{cases} O(n) & \text{lista semplice} \\ O(1) & \text{lista doppia} \end{cases}$

4. Lista ordinata:

Ricerca	**Inserimento**	**Cancellazione**
$O(n)$	Ordinato - $O(n)$	$\begin{cases} O(n) & \text{lista semplice} \\ O(1) & \text{lista doppia} \end{cases}$

2.1 Tabelle Hash

Una implementazione migliore sono le *tabelle hash*.

2.1.1 Tavole ad indirizzamento diretto

Prima di parlare delle tabelle hash, ha senso introdurre le *tavole ad indirizzamento aperto*. Sia U l'*universo delle chiavi*, ossia l'insieme di tutte le chiavi che possono essere inserite nel dizionario. Si prenda un array di lunghezza $|U|$, e si inserisca semplicemente l'elemento di chiave k nella cella k-esima dell'array.

In questo modo tutte e tre le operazioni di dizionario possono essere eseguite in tempo costante.

L'inghippo: Se l'universo delle chiavi non fosse finito? Se esistessero più elementi con la stessa chiave?

2.1.2 Funzione hash

Per risolvere il primo problema consideriamo un array T di lunghezza m; possiamo introdurre una funzione $h : U \to \{0, 1, \ldots, m-1\} \subset \mathbb{N}$, detta funzione hash, che mappa ogni chiave su una delle m celle. Solitamente una funzione hash ha la forma:

$$h(k) = k \mod m$$

Rimane (peggiora) il problema delle *collisioni*, ossia delle chiavi mappate sulla stessa cella; h, se $m < |U|$, non può infatti essere iniettiva.

2.2 Hashing con chaining

Una possibile soluzione è far puntare ogni elemento dell'array T alla testa di una lista "di trabocco". Gli elementi di chiave k saranno inseriti in testa alla lista $T[h(k)]$.
In questo modo l'inserimento costa $O(1)$. Analizziamo invece la ricerca al caso medio.

2.2.1 Ricerca senza successo

Sia m il numero di celle della tabella, n il numero di elementi in essa contenuti.

La ricerca **senza successo** di un elemento di chiave k comporta la scansione di tutta la lista $T[h(k)]$. Questa ha lunghezza, nel caso medio, è uguale ad $\dfrac{n}{m}$, il *fattore di carico* α della tabella hash. Considerando anche il costo del calcolo di $h(k)$, la complessità in tempo al caso medio della ricerca senza successo è $\Theta(1 + \alpha)$.

2.2.2 Ricerca con successo

La ricerca **con successo** di un elemento x di chiave k comporta la scansione di tutti gli elementi che sono stati inseriti nella lista $T[h(k)]$ dopo x.

Ipotiziamo (**Hashing uniforme semplice**) che ogni chiave abbia la stessa probabilità di essere associata ad una delle m celle di T.
Indichiamo con x_i l'i-esimo elemento inserito nella tavola; sia $k_i = x_i.key$. Definiamo la variabile aleatoria:

$$I\{h(k_i) = h(k_j)\} = \begin{cases} 0 & \text{se } h(k_i) \neq h(k_j) \\ 1 & \text{se } h(k_i) = h(k_j) \end{cases}$$

Nell'ipotesi di HUS, abbiamo che $\Pr\{h(k_i) = h(k_j)\} = 1/m \implies E[X_{ij}] = 1/m$ (ossia $1 \cdot 1/m + 0$).

Cerchiamo il **numero atteso di elementi inseriti nella lista dopo** x, uguale alla **media dei numeri attesi di elementi inseriti nella lista dopo** x_i, per $0 < i \leq n$.
Il numero atteso di elementi inseriti prima di x_i è uguale alla somma dei valori attesi di $I\{h(k_i) = h(k_j)\}$ per j compreso tra $i - 1$ ed n.

$$\sum_{j=i-1}^{n} E[I\{h(k_i) = h(k_j)\}] = \sum_{j=i-1}^{n} E[X_{ij}] = \sum_{j=i-1}^{n} \frac{1}{m} = \frac{n-i-1}{m}$$

Calcoliamo quindi la media, sugli n elementi della tavola, di $\dfrac{n-i-1}{m}$.

$$\begin{aligned} \frac{1}{n}\sum_{i=1}^{n} \frac{n-i-1}{m} &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{n}{m} - \frac{i}{m} - \frac{1}{m}\right) \\ &= \frac{1}{n}\left(\frac{n^2}{m} - \frac{n}{m} - \frac{1}{m}\sum_{i=1}^{n} i\right) \\ &= \frac{1}{n}\left(\frac{n^2}{m} - \frac{n}{m} - \frac{n(n+1)}{2m}\right) \\ &= \alpha - \frac{\alpha}{n} - \frac{\alpha}{2} - \frac{\alpha}{2n} \\ &= \frac{\alpha}{2} - \frac{\alpha}{2n} \end{aligned}$$

Considerando anche il tempo per esaminare l'elemento trovato, si ha un tempo medio di

$$1 + \frac{\alpha}{2} - \frac{\alpha}{2n} = \Theta(1 + \alpha)$$

L'importanza di questo risultato sta nel fatto che, quando $n = O(m)$, si ha che il costo della ricerca è costante.

2.3 Hashing con indirizzamento aperto

Un altro modello è quello dell'hashing con indirizzamento aperto. Gli elementi sono tutti memorizzati nell'array T, e le collisioni sono risolte tramite una *sequenza di ispezione*.
La funzione hash prende adesso due parametri – $h : U \times \{0, 1, ..., m-1\} \subset \mathbb{N} \to \{0, 1, ..., m-1\}$.
h è progettata in modo che $\langle h(k, 1), ..., h(k, m-1)\rangle$ sia una permutazione di $\langle 0, 1, ..., m-1\rangle$.

Ad ogni inserimento, partendo da $i = 0$, se la cella $h(k, i)$ è occupata si controlla la cella $h(k, i+1)$. Se questa è libera si inserisce, se no si incrementa i e si ricontrolla.
Una strategia simile si usa per la ricerca; si segue la sequenza di ispezione fino a trovare la chiave cercata.

2.3.1 Ricerca senza successo

Ipotizziamo (**Ipotesi di Hashing Uniforme**) che ogni chiave abbia la stessa probabilità di avere come sequenza di ispezione una delle $m!$ permutazioni di $\{0, 1, ..., m-1\}$.

In una ricerca senza successo ogni ispezione accede ad una cella occupata da un elemento di chiave diversa da k.
Definiamo la variabile aleatoria X come il numero di ispezioni fatte in una ricerca senza successo.
Un limite superiore per il valore atteso di X è

$$\begin{aligned}
E[x] &= \sum_{i=1}^{+\infty} i \cdot \Pr\{X = i\} = \sum_{i=1}^{+\infty} \Pr\{X \geq i\} \\
&= \sum_{i=1}^{+\infty} \text{ Probabilità di fare almeno } i \text{ accessi} \\
&= \sum_{i=1}^{+\infty} \text{ Probabilità di trovare le prime } i \text{ celle occupate} \\
&= \sum_{i=1}^{+\infty} \left(\frac{n}{m} \cdot \frac{n-1}{m-1} \cdot \ldots \cdot \frac{n-(i-2)}{m-(i-2)} \right) \\
&\leq \sum_{i=1}^{+\infty} \frac{n}{m}^{i-1} = \sum_{i=1}^{+\infty} \alpha^{i-1} = \sum_{i=0}^{+\infty} \alpha^{i} = \frac{1}{1-\alpha}
\end{aligned}$$

2.3.2 Ricerca con successo

La ricerca di una chiave k segue la stessa sequenza di ispezione che era stata seguita al suo inserimento.
Se k era la $i+1$-esima chiave inserita nella tabella hash, il numero atteso di operazioni fatte in una ricerca di k è al più $\dfrac{1}{1 - \underbrace{\frac{i}{m}}_{\alpha}} = \dfrac{m}{m-i}$.

Calcoliamo la media su tutte le n chiavi, ed otteniamo il numero medio di ispezioni durante una ricerca con successo.

$$\begin{aligned}
\frac{1}{n} \sum_{i=0}^{n-1} \frac{m}{m-i} &= \frac{m}{n} \sum_{i=0}^{n-1} \frac{1}{m-i} = \frac{1}{\alpha} \left[\frac{1}{m} + \frac{1}{m-1} + \ldots + \frac{1}{m-(n-1)} \right] \\
&= \frac{1}{\alpha} \sum_{t=m-n+1}^{m} \frac{1}{t} \leq \frac{1}{\alpha} \int_{m-n}^{m} \frac{1}{x} dx = \frac{1}{\alpha} \ln \frac{1}{1-\alpha}
\end{aligned}$$

2.4 Tipi di ispezione

2.4.1 Ispezione lineare

L'ispezione lineare prevede una funzione hash del tipo:

$$h(h'(k), i) = (k + i) \mod m$$

Dove h' è una funzione hash $U \rightarrow \{0, 1, \ldots m - 1\}$ ausiliaria.

Questo tipo di ispezione porta ad un fenomeno noto come *clustering primario*: Se una cella $T[j]$ è occupata, la chiave viene inserita nella successiva (se libera). Se si riprova ad inserire in $T[j]$, o in $T[j + 1]$, queste saranno trovate occupate, quindi si andrà ad inserire in $T[j + 2]$, e così via: in questo modo si crea un **agglomerato di celle adiacenti occupate**, che peggiorano le prestazioni.

Inoltre, ci sono solo m possibili sequenze, a fronte delle $m!$ permutazioni di m elementi.

2.4.2 Ispezione quadratica

Un altro tipo di ispezione è quella quadratica. La funzione hash ha la forma:

$$h(k, i) = (h'(k) + c_1 i + c_2 i^2) \mod m$$

Si noti come il punto di inizio della sequenza dipende solo da k, e non da i:
Ogni coppia di sequenze tali che $h(k_1, 0) = h(k_2, 0)$ saranno uguali in ogni altra posizione. Questo fenomeno si dice *clustering secondario.*

Come nell'ispezione lineare, ci sono solo m sequenze possibili.

2.4.3 Doppio Hashing

Una soluzione al problema del clustering è utilizzare due funzioni hash ausiliarie:

$$h = (h_1(k) + i h_2(k)) \mod m$$

In questo modo, a patto che $h_2(k)$ e m siano coprimi, sia la posizione iniziale che il "passo" dipendono da k, e ci sono m^2 possibili sequenze, che è sempre molto meno di $m!$, ma uno si accontenta.