# Tabelle e dizionari

Funzioni interessanti?

- Inserimento di un certo oggetto
- find di un certo oggetto
- delete

Lista: O(1), O(n), O(n)

Heap: O(log n), O(n)

Albero binario di ricerca: O(log n), O(log n)

Tabella: O(1), O(1) OTTIMO!!

Struttura molto brava a trovare uno specifico elemento, ma non in tutto. Per esempio potrei avere una struttura che ha pochissimi inserimenti e poi vive sulle query, quindi a me interessa quell'aspetto e non il resto.

Se sulla carta abbiamo una struttura che costa sempre O(n), ma un'operazione specifica costa O(1) allora potrebbe convenire rispetto alla tabella.

Le tabelle possono essere viste come degli array con degli indici Le tabelle hanno sono strutturate sulle coppie chiave-valore, {'ABC' : dati}. Concetto fondamentale per le tabelle di hash: Come passo dalla chiave ad un indice dell'array (che conterrà il valore)?

value = H[funzione.hash(key)] e questa funzione deve lavorare in tempo costante.

## Collisione

2 Chiavi diverse producono lo stesso indice.

# Funzioni di Hash

Quello che si vuole fare è generare un output con distribuzione uniforme partendo da una quasiasi stringa

Funzione di hash più semplice che si può fare:

```
int f (stringa s){
    int h = 0;

    for i -> lengt(s)
    h = (h+s[i])%256;

}
```

f('A') = 65 f('B') = 66

f('AA') = 130 f('AB') = 131

f('BA') = 131 Collisione

### diersi metodi di hashing

- Metodo della divisione
- Metodo della moltiplicazione
- Universal Hashing

# Algoritmi SHA

Anche soltanto 1 bit di differenza mi fa cambiare molto l'output della funzione di hashing (concetto i randomico)