Il problema della ricerca

▼ ***INDICE***

4 Il problema della ricerca

Nell'informatica troviamo problemi abbastanza ricorrenti, uno di questi è la ricerca di un elemento in un insieme di dati(ad es.numeri, cognomi, ecc…)

Questi problemi consistono in:

- Input: un array A ed un valore v da cercare al suo interno;
- Output: un indice i tale che $A[i] = v$, oppure $NULL(o-1)$ se il valore v non è presente nell'array.

4.1 Ricerca sequenziale

Un semplice algoritmo di ricerca è basato su:

- Ispezione: ossia controllare uno alla volta gli elementi dell'array;
- Confrontare: ossia confrontare ciascun elemento con v;
- Restituzione del risultato: interrompendosi appena(NON) trovato v.

Un esempio di ricerca sequenziale può essere questo indice:

```
def Ricerca_sequenziale(A,v):
  i = 0
  while((i<len(A)) and (A[i]!=v)):
    i+=1
  if i<len(A): return i
  else: return -1
```

Anche senza analizzare il codice possiamo vedere che il caso migliore è quando v si trova in $A[0]$ e quindi esce immediatamente dal ciclo, mentre il caso peggiore è il caso in cui v non si trova in A(poiché comunque dovrebbe venire analizzata l'intera lista per dire che l'elemento non c'è).

Dunque:

- Caso migliore: se $v = A[0]$, allora $\Theta(1)$;
- Caso peggiore: se $v \notin A$, allora $\Theta(n)$.

Visto che non abbiamo trovato una stima del costo che sia valida per tutti i casi, diremo che il costo computazionale dell'algoritmo è $\Theta(n)$.

4.2 Stima del costo medio

Quando il costo migliore e peggiore sono diversi, non è possibile determinare un valore stretto per il costo computazionale, possiamo però domandarci quale sia il costo computazionale dell'algoritmo nel caso medio.

Immaginiamo di avere un array A di n elementi al cui interno ogni posizione ha la stessa probabilità di contenere il valore v da cercare.

Possiamo dire che la probabilità che v sia in k-esima posizione è:

$$P = \frac{1}{n}$$

Applicando tale probabilità al numero totale di iterazioni, otteniamo:

$$P \cdot \sum_{k=0}^{n} k = \frac{1}{2} \cdot \frac{n(n+1)}{2} = \frac{n+1}{2}$$

Dunque possiamo dire che in media il ciclo viene eseguito $\frac{n+1}{2}$ volte, ossia $\Theta(n)$. Quindi il caso medio, si avvicina più al caso peggiore.

Il costo medio può essere trovato anche con il calcolo delle permutazioni.

Come ben sappiamo le permutazioni di una lista di n elementi corrisponde a $n!$, quindi possiamo dire che le permutazioni totali di A sono:

$$P_{tot} = n!$$

Dove al suo interno vi sono anche i seguenti sottoinsiemi:

- Permutazioni con v in prima posizione;
- Permutazioni con v in seconda posizione;
- Permutazioni con v in terza posizione;
- …

Tali sottoinsiemi, corrispondono ad una permutazione di $n-1$ elementi, ossia:

$$P_k = (n-1)!$$

quindi:

$$\sum_{k=0}^{n} k \cdot \frac{P_k}{P_{tot}} = \sum_{k=0}^{n} k \cdot \frac{(n-1)!}{n!} = \frac{1}{n} \sum_{k=0}^{n} k = \frac{n+1}{2}$$

Come possiamo vedere anche in questo verrà $\frac{n+1}{2}$ ossia $\Theta(n)$.

4.2.1 Operatore *in del linguaggio Python*

Prima di parlare dell’operatore *in* del linguaggio Python, bisogna ricordare la sua struttura:

$$< VALORE > IN < LISTA >$$

Ed un esempio del suo utilizzo può essere:

```
if v in A:
  print("Il valore v si trova in A")
else:
  print("Il valore v non si trova in A")
```

A prima vista essendo una condizione diremo che il costo è $\Theta(1)$, in realtà non è così, poiché esso corrisponde ad una ricerca sequenziale sappiamo benissimo che il costo sarà $\Theta(n)$.