Paolo

Paolo

Maggio 2023

Indice

Lista di algoritmi

1 Ricerca

Trovare un elemento k in un array ordinato in tempo $O(\log n)$ tramite il paradigma Divide et Impera.

Relazione di ricorrenza:

$$T(n) \leq \begin{cases} c_0 & \text{se } n \leq 1 \text{ oppure } k \text{ è l'elemento centrale} \\ T(n/2) + c & \text{altrimenti} \end{cases}$$

Algoritmo:

```
1:  RicercaBinariaRicorsiva(a, k, sinistra, destra):
2:    if (sinistra > destra) then
3:      return -1
4:    c = (sinistra + destra) / 2
5:    if k == a[c] then
6:      return c
7:    if sinistra == destra then
8:      return -1
9:    if (k < a[c]) then
10:     return RicercaBinariaRicorsiva(a, k, sinistra, c-1)
11:   else
12:     return RicercaBinariaRicorsiva(a, k, c+1, destra)
```

Algoritmo 1.1: Ricerca binaria ricorsiva

Algoritmo 1.1: nota[1].
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

[1] Forza paolo

1.1 Esempio di utilizzo

Trovare l'ultima occorrenza di un elemento x in un array ordinato in tempo $O(\log n)$ tramite il paradigma Divide et Impera.

Algoritmo:

```
1:  UltimaOccorrenza(a, l, r, x):
2:    if l > r then
3:      return -1
4:    if l == r ∧ a[l] == x then
5:      return l
6:    else
7:      return -1
8:    c = (l + r) / 2
9:    if a[c] ≤ x then
10:     return max(c, UltimaOccorrenza(a, c + 1, r, x))
11:   else
12:     return UltimaOccorrenza(a, l, c - 1, x)
```

Algoritmo 1.2: Ultima occorrenza

Algoritmo 1.2: nota[2].
Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

[2] Forzaa paolo

2 Interval Scheduling Pesato

Risoluzione del problema dell'Interval Scheduling Pesato.

Relazione di ricorrenza:

$$OPT(j) = \begin{cases} 0 & \text{se } j = 0 \\ \max(v_j + OPT(p(j)), OPT(j-1)) & \text{altrimenti} \end{cases}$$

Esecuzione esterna:

```
1: Input: n, s_1, ..., s_n, f_1, ..., f_n, v_1, ..., v_n
2: Ordina i job per tempi di fine in modo che f_1 ≤ f_2 ≤ ··· ≤ f_n
3: Calcola p(1), p(2), ..., p(n)
4: for j = 1 to n do
5:    M[j] = ∅ {Array globale}
```

Algoritmo:

```
1: M-Compute-Opt(j):
2:    if j = 0 then
3:       return 0
4:    if M[j] = ∅ then
5:       M[j] = max(v_j + M-Compute-Opt(p(j)), M-Compute-Opt(j − 1))
6:    return M[j]
```

Algoritmo 2.1: M-Compute-Opt

Algoritmo 2.1: nota[1].

Teorema 2.1 (Correttezza di M-Compute-Opt)**.** *L'algoritmo computa correttamente OPT(j)*

Dimostrazione. Per induzione.

Caso base $j = 0$. Il valore restituito è correttamente 0.

Passo induttivo. Consideriamo un certo $j > 0$ e supponiamo (ipotesi induttiva) che l'algoritmo produca il valore corretto di $OPT(i)$ per ogni $i < j$. Il valore computato per j dall'algoritmo è:

M-Compute-Opt$(j) = \max(v_j$+M-Compute-Opt$(p(j))$, M-Compute-Opt$(j-1))$

Siccome per ipotesi induttiva:

- Valore computato da M-Compute-Opt$(p(j)) = OPT(p(j))$ e
- Valore computato da M-Compute-Opt$(j-1) = OPT(j-1)$

Allora ne consegue che:

M-Compute-Opt$[j] = \max(v_j + OPT(p(j)),\ OPT(j-1)) = OPT(j)$ $\square$

[1] Forzaaa paolo