Azzolini Riccardo 2019-03-20

Eliminazione della ricorsione

1 Classificazione della ricorsione

La **ricorsione** si suddivide in

diretta: una procedura/funzione `F` chiama se stessa direttamente;

indiretta: una procedura/funzione `F` ne chiama un'altra `G`, e a sua volta `G` chiama `F`.

2 Eliminazione della ricorsione in coda

La **ricorsione in coda** (o **ricorsione terminale**) è un caso particolare di ricorsione nel quale la chiamata ricorsiva è l'ultima istruzione della funzione.

```
void F(x) {
    if P(X) { D; }
    else { E; y = g(x); F(y); }
}
```

Essa si può sostituire con un costrutto iterativo,[1] senza bisogno di strutture dati aggiuntive:

```
void F_it(x) {
    while (!P(x)) { E; x = g(x); }
    D;
}
```

[1] Molti compilatori effettuano automaticamente questa sostituzione.

2.1 Esempio: ricerca binaria

La **ricerca binaria** (o **dicotomica**), ad esempio di un oggetto in un vettore, può essere implementata mediante ricorsione in coda:

```
public int binsearch(int sx, int dx, Item el) {
    if (sx > dx) return -1;
    int x = (sx + dx) / 2;
    if (el == a[x]) return x;
    if (el < a[x]) return binsearch(sx, x - 1, el);
    return binsearch(x + 1, dx, el);
}
```

Di conseguenza, si può ricavare la corrispondente versione iterativa, trasformando i parametri `sx` e `dx` in variabili locali e la ricorsione in un ciclo:

```
public int binsearch_it(Item el) {
    int sx = 0;
    int dx = size - 1;
    while (sx <= dx) {
        int x = (sx + dx) / 2;
        if (el == a[x]) return x;
        if (el < a[x]) dx = x - 1;
        else sx = x + 1;
    }
    return -1;
}
```

2.1.1 Complessità

Nel caso migliore, cioè se l'elemento cercato si trova in mezzo al vettore, viene effettuato un solo confronto, quindi la ricerca richiede tempo $O(1)$ e spazio $O(1)$ in entrambe le implementazioni.

Il caso peggiore si ha invece quando l'elemento cercato è assente. Siccome la dimensione della parte di vettore considerata si dimezza a ogni passo, il numero di confronti effettuati è asintotico alla soluzione dell'equazione di ricorrenza

$$c_1 = 1, \quad c_n = 1 + c_{\frac{n}{2}}$$

dove n è la dimensione del vettore. Supponendo $n = 2^k \iff k = \log_2 n$, si ottiene

$$
\begin{aligned}
c_{2^k} &= 1 + c_{2^{k-1}} \\
&= 1 + 1 + c_{2^{k-2}} \\
&\vdots \\
&= 1 + 1 + \cdots + c_{2^{k-k}} \\
&= 1 + 1 + \cdots + c_1 \\
&= 1 + \underbrace{1 + \cdots + 1}_{k} = k + 1
\end{aligned}
$$

Allora, nel caso peggiore, si effettuano $\Theta(\log n)$ confronti, e di conseguenza:

- il tempo di calcolo è $\Theta(\log n)$ per entrambe le implementazioni, perché ogni confronto richiede tempo $O(1)$;
- lo spazio necessario è $\Theta(\log n)$ per la versione ricorsiva, a causa dei record di attivazione, e $O(1)$ per quella iterativa, che usa solo alcune variabili locali.

Ricapitolando, la ricerca binaria in un vettore di n elementi richiede in generale tempo $O(\log n)$ e spazio

- $O(\log n)$ per la versione ricorsiva;
- $O(1)$ per la versione iterativa.