Tutorato 04 Programmazione

Giulio Umbrella

Contents

1 Ricorsione

La **ricorsione** e' una tecnica di programmazione in cui una funzione chiama se stessa per risolvere un problema utilizzando un problema di taglia piu' piccola.

Per dimostrare che una funzione ricorsiva e' corretta usiamo la dimostrazione per **induzione**

La ricorsione ha **sempre** almeno un caso base

La chiamata ricorsiva e' sempre fatta alla stessa funzione ma con un problmea di taglia piu' **piccola**

Al termina della chiamata ricorsiva combino il risultato e restituisco al chiamante

1.1 Programmazione ricorsiva

Le funzioni ricorsive hanno **due** opzioni mutualamente esclusive.

1. Caso base - uno o piu'
2. Chiamata ricorsiva ad un sotto problema

Quando affronto le funzioni ricorsive devo sempre

1. Determinare se mi trovo nel caso base oppure se devo fare una chiamate ricorsiva
2. Determinare il sotto problema che devo risolvere - e gli eventuali parametri da passare.

Il caso base e' la soluzione di un problema elementare - solitamente basta restituire un singolo valore o eseguire operazioni elementari

La chiamata ricorsiva ha due solitamente tre passi

1. Eseguire la chiamata ricorsiva
2. Combinare il valore ottenuto con altre informazioni
3. Restituire un valore al chiamante

1.2 Dimostrazione

La dimostrazione per induzione e' divisa in **due** parti distinte

1. **Caso base**: Dimostrare che il caso base e' giusto
2. **Passo induttivo**: assumere che una instanza del sottoproblema e' giusto e dimostare che puo' essere usata per risolvere il problema.

Informalmente, possiamo pensare alla chiamata ricorsiva come ad una "scatola magica" che risolve il sottoproblema. Ovviamente dobbiamo capire il giusto input da fornire e come utilizzare l'output che viene fornito.

2 Esempio

Dimostriamo per induzione che possiamo fare cadere le tessere di un domino.

1. Caso base: Posso fare cadere la prima tessera
2. Passo induttivo: Se faccio cadere una tessera del domino, quella successiva cade

Applicando questo schema, so che tutte le tessere cadono.

3 Esercizio conta valori

Scrivere una funzione ricorsiva che dato un array di interi conta le occorrenze di un intero x

3.1 Implementazione

```
/*
PRE: un array di interni dimensione maggiore o uguale a zero
POST: il numero di occorrenze di x all'interno dell'array
*/
int conta_valori(a,x,n){
    // caso base
    if( n == 0)
        return 0;

    // chiamata ricorsiva
    if a[0] == x
        return 1 + cv(a++,x,n-1)
    else
        return cv(a++,x,n-1)
}
```

3.2 Dimostrazione

Caso Base Se l'array e' vuoto - n = 0 - non ci possono essere occorrenze di x, quindi restituisco zero.

Passo induttivo

Ipotesi induttiva:

Al termine della chiamata ricorsiva la funzione mi restituisce il numero di occorrenze di x nella *parte restante* dell'array. Quello che devo fare e' controllare il primo valore dell'array per verificare se e' uguale a x.

4 Esercizio somma cifre

Scrivere una funzione ricorsiva che dato un array in input, somma le sue cifre. Eg con 1987 in input 25 = 1+9+8+7 .

4.1 Risoluzione problema

Per prima cosa ricordiamo che possiamo esprimere ogni numero intero a rispetto alla divisione con un intero b in questa forma

`a = b*q + r`

Partiamo da un semplice esempio e prendiamo 1987.

```
1987 = 198 * 10 + 7
198  = 19  * 10 * 8
19   = 1   * 10 + 9
1    = 0   * 10 + 1
0    = 0   * 10 + 0
```

Quindi possiamo ottenere la soluzione come somma dei resti delle operazioni di divisione. Fermiamo le operazioni quando il resto diventa zero.

4.2 Implementazione

```
somma_cifre_rec(int n)
{
    if(n == 0)
        return 0;

    q = n / 10; // quoziente
    r = n % 10; // resto

    return r + somma_cifre_rec(q);
}
```

4.3 Dimostrazione

Caso base

Se l'intero n e' 0, la somma delle sue cifre e' 0. Quindi il caso base e' corretto.

Passo induttivo

Per prima cosa

Ipotesi induttiva: dato un numero intero n di k cifre, la funzione ricorsiva restituisce la somma delle cifre passando come input il numero di k - 1 cifre.

Quindi come primo passo, scriviamo l'intero in input come n = 10 * q + r. A questo punto sappiamo che

- r e' la prima cifra del valore n
- somma_cifre_rec(q) restituisce la somma delle cifre

Quindi se sommiammo r con somma_cifre_rec(q) otteniamo il valore corretto.

Esempio prendiamo il valore di input 1987 e scriviamo come 1987 = 198 * 10 + 7. La funzione somma_cifre_rec(198) restituisce il corretto valore per ipotesi induttiva e otteniamo 18. Aggiungiamo il valore 7 e otteniamo 25.

5 Esercizio stampa base 2

Scrivere un programma ricorsivo che dato in input un intero n stampa tutte i valori in base 2 compresi fra 0 e n - 1. Ad esempio se n = 2, stampa

```
11
10
01
00
```

5.1 Implementazione

```c
#include <stdio.h>
#include <string.h>

void copyArray(char dst[],char src[]);
void print_bin_rec(char s[],int k);
int atoi(const char *str);

int main(int argc, char ** argv)
{
    int i = atoi(argv[1]);
    char s[] = {'\0'};

    print_bin_rec(s,i);
}

void copyArray(char dst[],char src[])
{
    while((*dst++ = *src++) != '\0')
        ;
}

void print_bin_rec(char s[], int k)
{

    // Caso base
    if(k == 0)
        printf("%s\n",s);

    else{
        int s_len = (int) strlen(s);

        char s1[s_len + 1 + 1];
        char s0[s_len + 1 + 1];

        // copy prefix
        copyArray(s1,s);
        copyArray(s0,s);

        // Set last  bit
        s1[s_len] = '1';
        s0[s_len] = '0';

        // Set null byte
        s1[s_len + 1] = '\0';
        s0[s_len + 1] = '\0';

        // recursive call
        print_bin_rec(s1, k-1);
        print_bin_rec(s0, k-1);
```

```
    }

}
```

6 Esercizi aggiuntivi

1. Scrivere la prova induttiva dell'esercizio stampa base 2
2. Scrivere un programma simile all'esercizio stampa base 2 che stampi i valori in formato *big endian**. Ad esempio con k = 2 deve stampare:

```
11
10
01
00
```

I bit piu' significativi appaiono sulla sinistra.

3. Scrivere un programma analogo all'esercizio stampa base 2 che funzioni in base 16.