

Informatica 1

Corso di Laurea Triennale in Matematica

Gianluca Rossi

`gianluca.rossi@uniroma2.it`

Dipartimento di Matematica
Università di Roma "Tor Vergata"

13: QuickSort su liste



Ordinare una lista di elementi

Ordinamento

Data una lista L di elementi (p.e. interi), si vuole modificare la lista in modo che la lista risultante risulti ordinata (p.e. in modo non decrescente).



Quick Sort

- Si seleziona un elemento della lista detto *pivot*;
- Si modifica la lista in modo che tutti gli elementi più piccoli del *pivot* lo precedano e tutti quelli più grandi lo seguano.
- Ricorsivamente si esegue l'algoritmo sulla porzione di lista composta da tutti gli elementi che precedono il pivot.
- Ricorsivamente si esegue l'algoritmo sulla porzione di lista composta da tutti gli elementi che seguono il pivot.



La lista da ordinare

```
struct listelem {  
    int info;  
    struct listelem *next;  
};  
typedef struct listelem listelem;  
listelem *L;
```



La funzione Quick sort

```
listelem *quicksort(listelem*, listelem*);
```

- Se l'elemento puntato dal primo puntatore precede l'elemento puntato dal secondo puntatore, la funzione ordina la sottolista delimitata dai due puntatori in input.
- Restituisce il puntatore al primo elemento della sottolista modificata.



Richiama una funzione **split** che:

- Seleziona il *pivot*: Il primo elemento della lista.
- Modifica la porzione di lista interessata in modo da far precedere il pivot da tutti gli elementi minori del *pivot*.
- Tutti gli altri elementi seguiranno il *pivot*.

```
listelem *split(listelem*, listelem*);
```



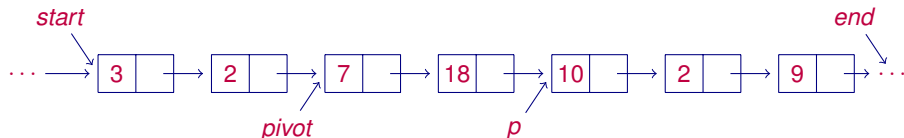
Quick sort

```
01: listelem *quicksort(listelem *start, listelem *end){  
02:   listelem *pivot;  
03:   if(start == end)  
04:     return start;  
05:   pivot = start;  
06:   start = split(start, end);  
07:   start = quicksort(start, pivot);  
08:   pivot->next = quicksort(pivot->next, end);  
09:   return start;  
10: }
```

- Ci si accerta che la lista contenga elementi (linea 03);
- Si memorizza il riferimento al *pivot* (linea 05);
- Si esegue lo *split* sul *pivot* selezionato (linea 06);
- Si richiama ricorsivamente la funzione di ordinamento sulle due sottoliste delimitate dal *pivot* (linee 07 e 08).



La funzione **split**



- *p* scorre da *start* \equiv *pivot* fino a *end*;
- tutti gli elementi che precedono *pivot* devono essere inferiori di *pivot*
- tutti gli elementi delimitati da *pivot* e *p* sono \geq di *pivot*
- quelli che seguono *p* devono essere processati



Quick sort

```
listelem *split(listelem *start, listelem *end){
    listelem *p, *tempstart, *tempp, *pivot;
    if ( start == NULL )
        return NULL;
    pivot = start;
    p = start;
    while( p->next != end){
        if( (p->next->info) < (pivot->info)){
            /* p->next deve essere messo in testa */
            tempp = p->next;
            tempstart = start;
            p->next = tempp->next;
            start = tempp;
            start->next = tempstart;
        } else
            p = p->next;
    }
    return start;
}
```



Quick sort

```
int main(int argn, char *args[]){
    listelem *L = NULL, *t;
    int i;
    /* creazione della lista */
    for(i = argn-1; i > 0; i--){
        t = L;
        L = (listelem*)malloc(sizeof(listelem));
        sscanf(args[i], "%d", &(L->info));
        L->next = t;
    }
    L = quicksort(L, NULL);
    printList(L);
}
```



Efficienza

Sia $T(n)$ il numero di operazioni elementari eseguite dal Quick sort su una lista di n elementi:

$$T(n) = \underbrace{c \cdot n}_{\text{la funzione slit}} + \underbrace{T(\ell) + T(n - \ell - 1)}_{\text{le due quicksort}}$$

Supponiamo $\ell \approx \frac{n}{2}$

$$\begin{aligned} T(n) &\approx cn + 2T\left(\frac{n}{2}\right) = cn + 2\left(c\frac{n}{2} + 2T\left(\frac{n}{4}\right)\right) = 2cn + 4T\left(\frac{n}{4}\right) \\ &= 3cn + 8T\left(\frac{n}{8}\right) = \dots = kcn + 2^k T\left(\frac{n}{2^k}\right) \end{aligned}$$

Il processo ha termine quando $k \approx \log(n)$

$$\begin{aligned} T(n) &\approx c \cdot n \cdot \log(n) + n \cdot T(1) = c \cdot n \cdot \log(n) + c' \cdot n \\ &\approx c \cdot n \cdot \log(n) \end{aligned}$$

