Stringhe palindrome

Una stringa si dice palindroma se è uguale alla sua trasposta, cioè se è identica se letta da sinistra e destra o da destra a sinistra: per esempio, "alla" e "ara" sono palindrome.
Scopo dell'esercizio è scrivere una funzione $f(s)$ che ritorna il numero minimo di caratteri da **inserire** in s necessari per rendere s palindroma.
Per esempio, quanti caratteri sono necessari per rendere palindroma la parola "casacca"?

- Una soluzione banale potrebbe essere quella di "specchiare" la parola, aggiungendo $n = 7$ caratteri: "casaccaACCASAC" (le lettere inserite sono scritte in maiuscolo).
- Migliorando un poco, è possibile specchiare la parola sull'ultima lettera: "casaccaCCASAC", che richiede $n - 1 = 6$ caratteri.
- Si potrebbe notare che "casacca" contiene "acca" che è già palindroma. Si può sfruttare questo fatto, inserendo 3 caratteri: "casaccaSAC";
- Infine, si potrebbe notare che "casac" è ancora palindroma, e quindi bastano due caratteri: "ACcasacca".

Notate che non necessariamente i caratteri si inseriscono in testa o in fondo; per esempio, "anta" può essere resa palindroma con un carattere: "antNa".
Fornire un algoritmo per calcolare $f()$ e stampare la stringa separata da spazi. Discutere la complessità dell'algoritmo.

È possibile notare che se il primo e l'ultimo carattere di una stringa sono uguali, è possibile "eliminarli" e concentrarsi sulla parte rimanente. Se invece sono diversi, si possono considerare due possibilità: rende palindromo il primo carattere oppure rendere palindromo l'ultimo.

- Se la stringa $s = \mathrm{a}s'\mathrm{a}$ è composta da due identici caratteri "a" iniziale e finale, allora:

$$f(s) = f(s')$$

- Se la stringa $s = \mathrm{a}s'\mathrm{b}$ ha due caratteri iniziale e finale diversi, aggiungiamo o un carattere "b" in testa (e consideriamo il problema $\mathrm{a}s'$, oppure aggiungiamo un carattere "a" in coda (e consideriamo il problema $s'\mathrm{b}$). Scegliamo fra le due possibilità quella con costo minore. In entrambi i casi, dobbiamo sommare 1 per il carattere aggiunto.

$$f(s) = \min\{f(\mathrm{a}s'), f(s'\mathrm{b})\} + 1$$

A questo punto, cerchiamo di risolvere il problema tramite memoization. Data una stringa $s[1 \dots n]$, utilizziamo una tabella di appoggio $F[1 \dots n, 1 \dots n]$.

La funzione calcolaF(s, i, j) calcola il costo per rendere palindroma la sottostringa $s[i \dots j]$. Il problema originale è ovviamente calcolaF(s, i, n).

```
calcolaF(ITEM[] s, integer[][] F, integer i, integer j)
if j ≤ i then
    return 0
else if F[i, j] = ⊥ then
    if s[i] = s[j] then
        F[i, j] ← calcolaF(s, i + 1, j − 1)
    else
        F[i, j] ← min(calcolaF(s, i, j − 1), calcolaF(s, i + 1, j)) + 1
    return F[i, j]
```

Una procedura per stampare la risultante stringa palindroma opera ricorsivamente. Se il primo e l'ultimo carattere sono uguali, richiama se stessa sulla parte centrale. Se sono diversi, individua il quale delle due possibilità è stata scelta e chiama se stessa nella parte opportuna.

```
stampa(ITEM[] s, integer[][] F, integer i, integer j)
if s[i] = S[j] then
    print s[i] + stampa(s, f, i + 1, j − 1) + s[j]
else if F[i, j] = F[i + 1, j] + 1 then
    print s[i] + stampa(s, F, i + 1, j) + s[i]
else
    print s[j] + stampa(s, F, i, j − 1) + s[j]
```

Inizializzazione, calcolo e stampa:

```
for i ← 1 to n do
    for j ← i + 1 to n do
        F[i, j] ← ⊥
calcolaF(s, F, 1, n)
stampa(s, F, 1, n)
```

Sulla complessità in tempo: $O(n^2)$, il tempo per riempiere tutta la tabella (nel caso peggiore).
Sulla complessità in spazio: uguale alla complessità in tempo, a meno che non si rinunci alla stampa e si memorizzi solo la riga precedente. Nel qual caso la complessità in spazio è $O(n)$.

Esercizio 13.1-5

Dimostrate che, in un albero rosso-nero, il percorso semplice più lungo che va da un nodo x ad una foglia discendente di x ha un'altezza al massimo doppia di quello del percorso semplice più breve dal nodo x a una foglia discendente.

Soluzione. Per le proprietà degli alberi rosso-neri tutti i cammini semplici da x ad una foglia discendente contengono lo stesso numero di nodi neri b. Poiché lungo un cammino semplice non possiamo avere due nodi rossi consecutivi, denotando con r il numero di nodi rossi in un cammino semplice da x ad una foglia discendente, abbiamo che $0 \leq r \leq b+1$. Pertanto, per la lunghezza $\ell = r + b$ di un cammino da un nodo x ad una foglia discendente, abbiamo

$$b \leq \ell \leq 2b + 1.$$

Possiamo quindi concludere che il rapporto tra il più lungo e il più breve cammino semplice da x ad una foglia discendente è al più $2 + 1/b$.

Esercizio 13.2-4

Dimostrate che ogni albero di ricerca di n nodi può essere trasformato in un qualsiasi altro albero binario di ricerca con n nodi effettuando $O(n)$ rotazioni.

Soluzione. Dimostriamo prima di tutto che bastano al più $n-1$ rotazioni destre R per trasformare un qualsiasi albero T di n nodi in una catena destra. Definiamo il *dorso destro* di un albero come l'insieme dei nodi che si trova sul cammino dalla radice alla foglia più a destra ed osserviamo che il dorso contiene sempre almeno un nodo (la radice). Notiamo che se effettuiamo una rotazione a destra su un nodo del dorso destro che ha un figlio sinistro, il numero di nodi nel dorso destro aumenta di uno. Quindi dopo $n-1$ rotazioni il dorso contiene n nodi (e quindi l'albero è una catena destra).

Supponiamo di voler trasformare l'albero T_1 nell'albero T_2. Sappiamo che esiste una sequenza R_1 di rotazioni destre che trasforma T_1 nella catena destra e una sequenza R_2 di rotazione destre che trasforma T_2 nella catena destra. Sia L_2 la sequenza che si ottiene da T_2 sostiutuendo ogni rotazione destra con la corrispondente rotazione sinistra ed invertendo l'ordine delle rotazioni. Osserviamo che se applichiamo la sequenza L_2 alla catena destra otteniamo T_2. Quindi applicando R_1 a T_1 otteniamo la catena destra ed applicando L_2 alla catena destra otteniamo T_2. Siccome sia R_1 che L_2 contengono $O(n)$ rotazioni abbiamo mostrato una sequenza di $O(n)$ rotazioni che trasforma T_1 in T_2.

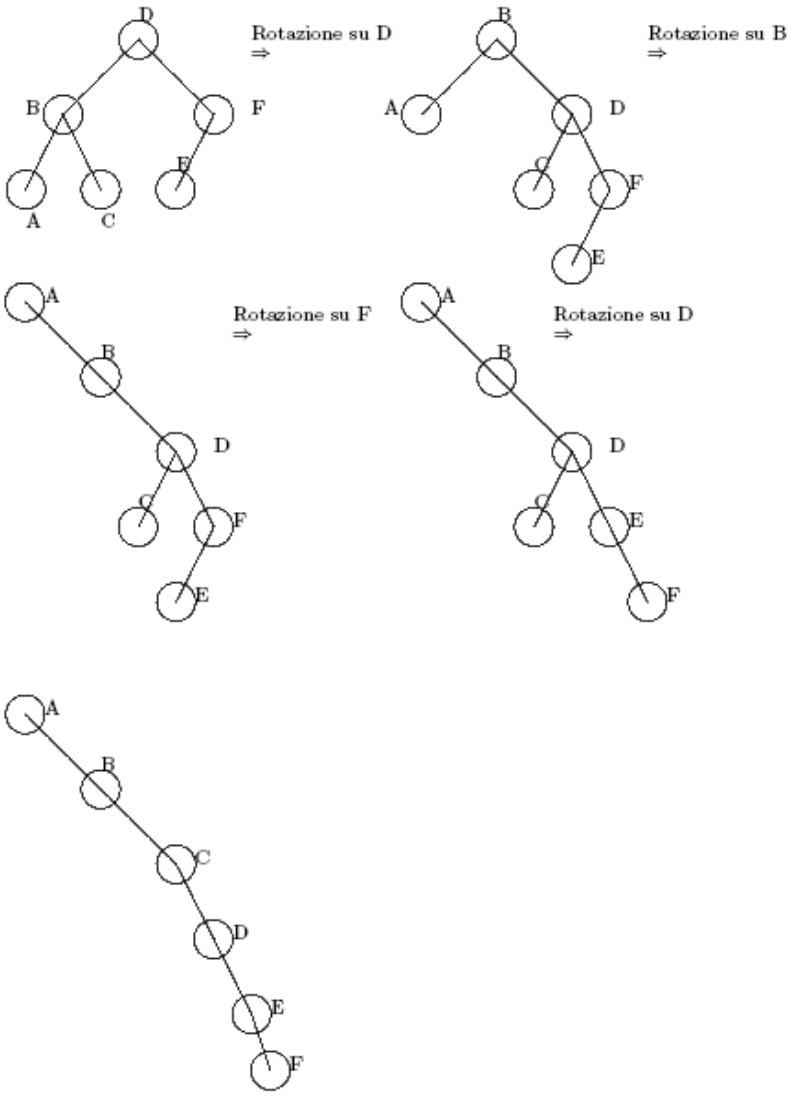

FIGURA 1. Un esempio di trasformazione di un albero in una catena destra mediante rotazioni a destra.