

Mosse su scacchiera (Esercizio 1.7 in 13-pd.pdf)

- Supponete di avere una scacchiera di dimensione $n \times n$ che rappresenta una scacchiera e un pedone che dovete muovere dall'estremità inferiore a quella superiore.
- Il pedone si può muovere (1) una casella in alto, oppure (2) una casella in diagonale alto-destra, oppure (3) una casella in diagonale alto-sinistra.
- Alla scacchiera è associata una matrice di profitto P ; quando una cella (r, c) viene visitata, si guadagna un profitto reale $P[r][c]$

Scrivere un algoritmo che restituisca il massimo profitto ottenibile partendo da una qualunque cella dell'estremità inferiore e raggiungendo una qualunque cella dell'estremità superiore, seguendo le regole appena descritte.

	1	2	3	4	5
1	6	7	4	7	<u>8</u>
2	7	6	1	1	<u>4</u>
3	3	5	7	<u>8</u>	2
4	2	6	<u>7</u>	0	2
5	7	3	5	<u>6</u>	1

Donald Trump (Compito 16/7/2016)

La Route 66 è una strada che collega Chicago a Los Angeles e che contiene un totale di n città. Nel suo prossimo tour elettorale Donald Trump (The Donald) ha deciso di seguire la Route 66 in una direzione, senza mai tornare indietro sui propri passi. The Donald non può tenere un comizio in ogni città, ma deve sceglierne un sottoinsieme.

Date due città i, j in cui terrà comizi, esse devono trovarsi ad una distanza superiore o uguale a D . La città i -esima si trova al miglio $m[i]$; quindi la distanza fra i e j è pari a $|m[j] - m[i]|$.

Seguendo queste regole, The Donald vorrebbe parlare al maggior numero di elettori. Si stima che al comizio nella città i -esima saranno presenti $e[i]$ elettori.

Donald Trump

Sorprendentemente, The Donald non ha grandi conoscenze informatiche, e ha chiesto a voi di risolvere il problema; in particolare, vorrebbe un algoritmo che restituisca il maggior numero di elettori che possono essere presenti seguendo le regole di cui sopra, dati un vettore di posizioni delle città $m[]$ e un vettore di numero di elettori $e[]$. Scrivere l'algoritmo e commentatene correttezza e complessità. Nel caso vi rifiutaste di risolvere il problema per conto di The Donald, riceverete +1 punto bonus.¹

¹Non è vero, è uno scherzo.

Palindroma (Esercizio 1.15 di 13-pd.pdf)

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.

Scrivere un algoritmo `minpal(ITEM[] s)` che ritorna il numero minimo di caratteri da **inserire** in `s` per rendere `s` palindroma.

Per esempio, input: “casacca”:

- $n = 7$ caratteri: “casaccaACCASAC”
- $n = 6$ caratteri: “casaccaCCASAC”
- $n = 3$ caratteri: “casaccaSAC”
- $n = 2$ caratteri: “ACcasacca”

Notate che non necessariamente i caratteri si inseriscono in testa o in fondo; per esempio, “anta” \rightarrow “antNa”.

Spoiler alert!

Mosse su scacchiera (Soluzione 2.7 in 13-pd.pdf)

Definiamo una matrice DP tale che $DP[r][c]$ rappresenta il massimo guadagno che si può ottenere partendo da una cella (r, c) e arrivando ad una cella della riga in alto ($r = 1$), dove r e c rappresentano la riga e la colonna, rispettivamente.

Il valore di ritorno corrisponderà al valore massimo dell'ultima riga ($r = n$). Per esercizio, scrivere un algoritmo che restituisca anche il percorso, e non solo il valore massimo.

$$DP[r][c] = \begin{cases} -\infty & c < 1 \text{ or } c > n \\ P[1][c] & r = 1 \text{ and } 1 \leq c \leq n \\ \max \begin{pmatrix} DP[r-1][c-1], \\ DP[r-1][c], \\ DP[r-1][c+1] \end{pmatrix} & 2 \leq r < n \text{ and } 1 \leq c \leq n \end{cases}$$

Mosse su scacchiera

```
int searchPath(int[][] P, int n)
```

```
int DP = new int[1...n][1...n]  
for c = 1 to n do                                % Copia la prima riga nel vettore DP  
    DP[1][c] = P[1][c]  
  
for r = 2 to n do  
    for c = 1 to n do  
        DP[r][c] =  $-\infty$   
        foreach d  $\in \{-1, 0, +1\}$  do  
            int newc = c + d  
            if  $1 \leq \textit{newc} \leq n$  then  
                DP[r][c] =  $\max(DP[r][c], DP[r-1][\textit{newc}] + P[r][c])$   
  
return  $\max(DP[n])$                                 % Restituisce il massimo dell'ultima riga
```

Donald Trump (Compito 16/7/2016)

$$DP[j] = \begin{cases} e[1] & j = 1 \\ \max\{DP[j-1], e[j] + \max_{i < j \wedge m[j]-m[i] \geq D} DP[i]\} & j > 1 \end{cases}$$

Donald Trump

```
int serveTheDonald(int[] m, int[] e, int n, int D)
```

```
int[] DP = new int[1...n]
```

```
DP[1] = e[1]
```

```
for j = 2 to n do
```

```
    DP[j] = DP[j - 1]
```

```
    int i = 1
```

```
    while i < j and m[j] - m[i] ≥ D do
```

```
        DP[j] = max(DP[j], e[j] + DP[i])
```

```
        i = i + 1
```

```
return DP[n]
```

Palindroma (Soluzione 1.15 di 13-pd.pdf)

- Se $s = as'a$ è composta da due identici caratteri “a” iniziale e finale, allora:

$$\text{minpal}(s) = \text{minpal}(s')$$

- Se $s = as'b$ ha due caratteri iniziale e finale diversi
 - o aggiungiamo o un carattere “b” in testa (e consideriamo il problema as' , eliminando virtualmente il carattere b),
 - oppure aggiungiamo un carattere “a” in coda (e consideriamo il problema $s'b$, eliminando virtualmente il carattere a).

Scegliamo fra le due possibilità quella con costo minore. In entrambi i casi, dobbiamo sommare 1 per il carattere aggiunto.

$$\text{minpal}(s) = \min\{\text{minpal}(as'), \text{minpal}(s'b)\} + 1$$

- Se s contiene un carattere solo o nessuno carattere, s è palindroma e bisogna rispondere 0.

Palindroma

Sia $DP[i][j]$ il numero di caratteri che è necessario inserire per rendere palindroma la stringa $s[i \dots j]$; può essere calcolato in modo ricorsivo nel modo seguente:

$$DP[i][j] = \begin{cases} 0 & i \geq j \\ DP[i+1][j-1] & i < j \text{ and } s[i] = s[j] \\ \max(DP[i+1][j], DP[i][j-1]) & i < j \text{ and } s[i] \neq s[j] \end{cases}$$

Palindroma

```
int minpal(ITEM[] s, int n)
```

```
int[][] DP = new int[1...n][1...n] = {-1}    % Initialized to -1
```

```
return minpalRec(s, DP, 1, n)
```

```
int minpalRec(ITEM[] s, int[][] DP, int i, int j)
```

```
if j ≤ i then
```

- └ **return** 0

```
else if DP[i][j] < 0 then
```

- └ **if** *s*[*i*] == *s*[*j*] **then**
 - └ *DP*[*i*][*j*] = minpalRec(*s*, *i* + 1, *j* - 1)
- └ **else**
 - └ *DP*[*i*][*j*] = min(minpalRec(*s*, *i*, *j* - 1), minpalRec(*s*, *i* + 1, *j*)) + 1

```
return DP[i][j]
```
