EDIT DISTANCE

Titolo nota 17/03/2016

X e Y $X = x_1 \dots x_n$ $Y = y_1 \dots y_m$

errori mismatch

blank in X che corrisponde a un carattere in Y = inserzione

blank in Y " " " " " X = cancellazione

riversione

X = ATLETA

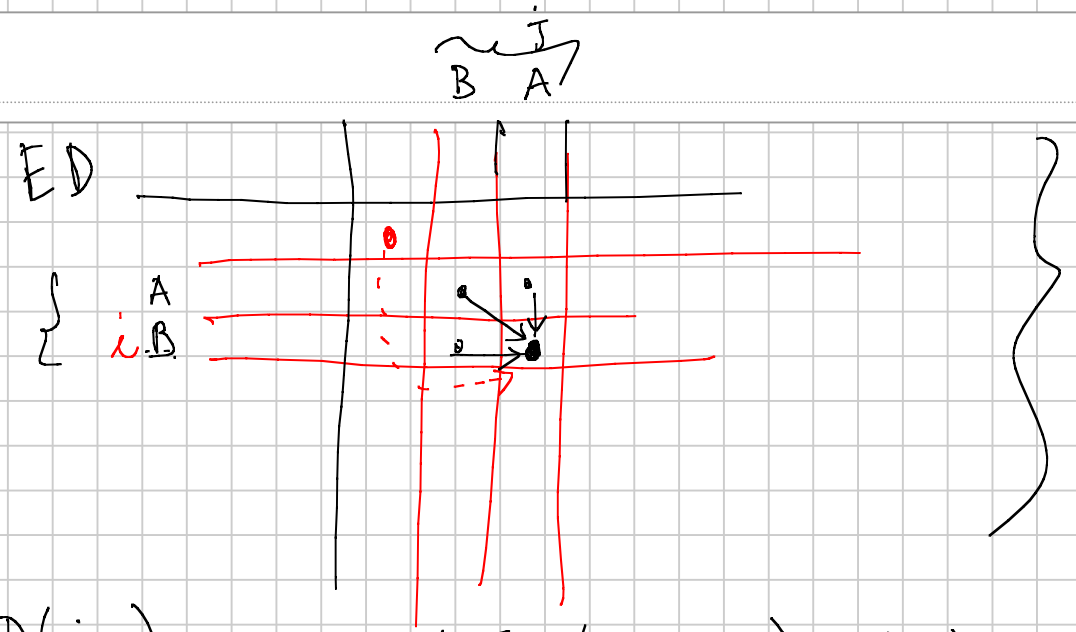
Y = ALTERA

X = ... ab .....

Y = ... ba .....

1 errore

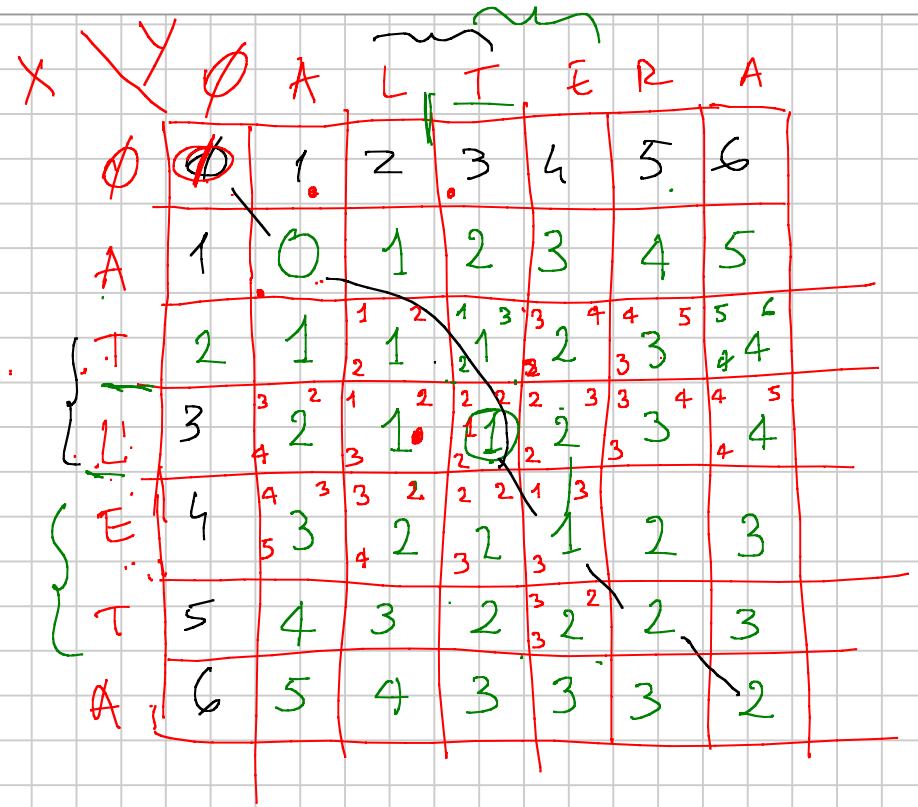
Titolo nota 17/03/2016



$$ED(i,j) = \min\left(ED(i-1,j-1) + p(i,j),\ ED(i,j-1)+1,\ ED(i-1,j)+1 \right)$$

$$ED(i-2,j-2)+1 \quad \text{if} \quad (x_i == y_{j-1} \ \&\& \ x_{i-1} == y_j$$

ED

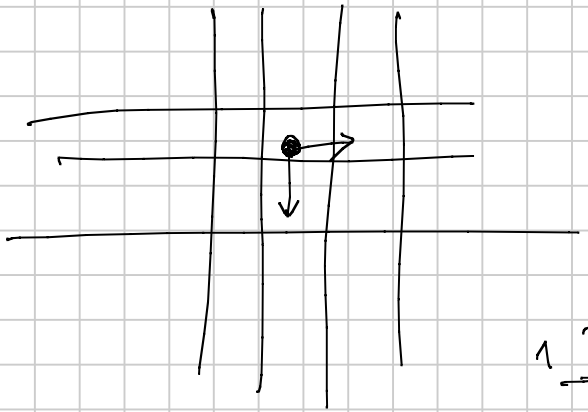


ATLETA
ALIERA
= 1 = 1 = 2 errori

AULTERA
ATLETUA
= ✗ = 1 1 = 3 errori

1 unica soluzione
ottimà con 2 errori

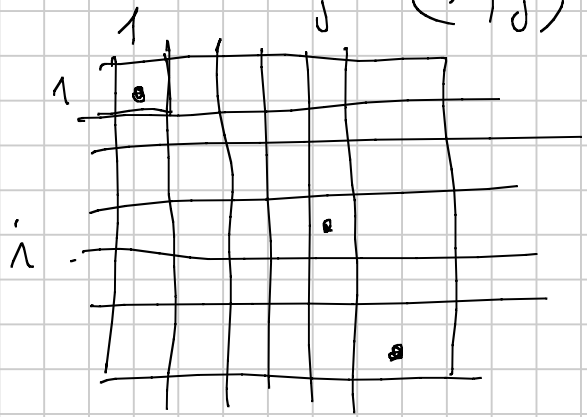
Scacchiera di $n \times n$ caselle

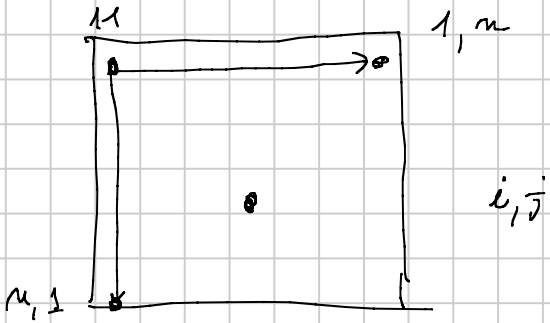


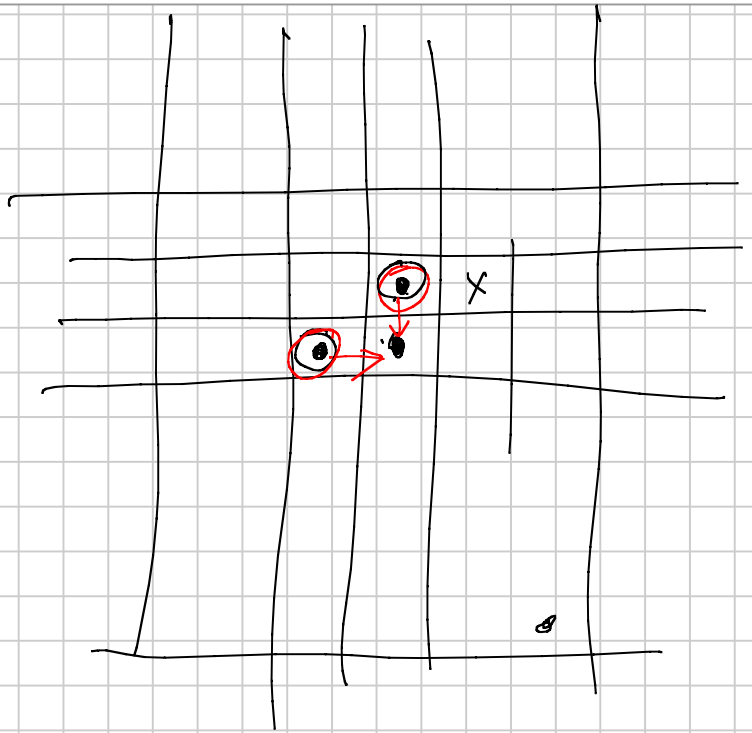
P.D.

$(i,j) \to (i+1, j)$, $i < n$

$(i,j) \to (i, j+1)$, $j < n$



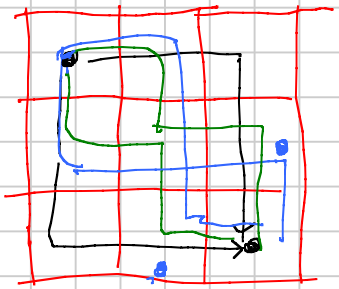




$P(i,0) = P(0,j) = 1$

$P(n,n)$

$P(i,j) = P(i-1,j) + P(i,j-1)$



$P(3,3) = 6$

```
Contapercorsi (n);
P = nuova matrice (n x n)
for i = 1 to n        P[i,1] = 1;
for j = 1 to n        P[1,j] = 1;
for i = 2 to n
    for j = 2 to n    P[i,j] = P[i-1,j] + P[i,j-1];
return P(n,n);
```

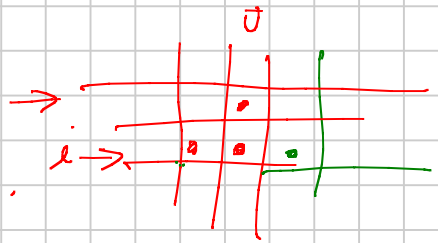
$\Theta(n^2)$ tempo

$\Theta(n^2)$ spazio

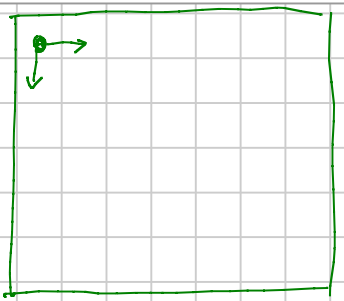
$\Downarrow$

$\Theta(n)$ memorizzando le ultime 2 righe

Memorizzare tutta la matrice non serve.



Spazio costante?

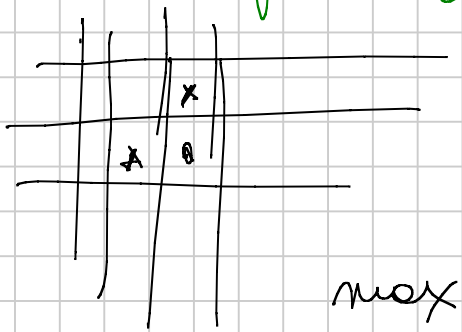


$(i,j) \to (i, j+1)$ a destra

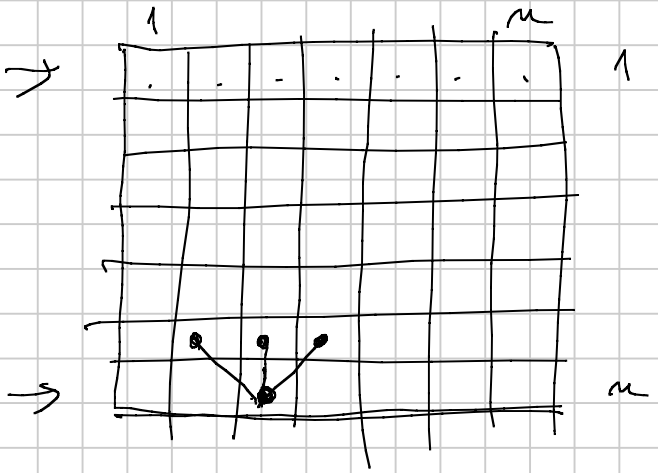
$(i,j) \to (i+1, j)$ in basso

C_{ij} si acquista un profitto c_{ij}

Alg. di PD che calcoli il percorso da $(1,1)$ a (n,n) col massimo profitto.



$$\max\left(S[i-1,j], S[i,j-1] \right) + c[i,j]$$



quando si visita x, y si prende un premio $\$(x,y) \geq 0$;

Calcolare il percorso da una qualunque casella dell'ultima riga a una qualunque casella della riga superiore che produce il profitto max, con un algoritmo di programmazione dinamica.

$S(x,y)$ il massimo profitto per arrivare a x, y

$S(1,y)$ " per arrivare alla riga 1

Regola ricorsiva

$$i < n \quad \begin{cases} S(n, j) = P(n, j) \\ S(i, j) = \max \left(S(i+1, j),\ S(i+1, j-1),\ S(i+1, j+1) + P(i, j) \right); \end{cases}$$

Ricerca cammino (P, n):

S = nuova array di dimensione n x n ;

M = for j = 1 to n S[n, j] = P[n, j] ;

for i = n-1 downto 1

  for j = 1 to n {

  S[i,j] = S [i+1, j] ; ↑ M[i,j] = 0

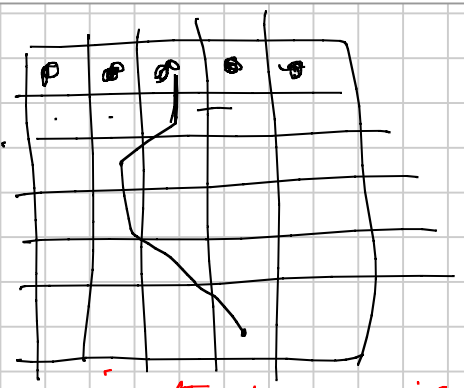
  if (j > 1) && S[i+1, j-1] > S[i,j]) S[i,j] = S[i+1, j-1] ; M[i,j] =

  if (j < n) && S(i+1, j+1] > S [i, j]) S [i,j] = S [i+1, j+1];

  M[i,j] =

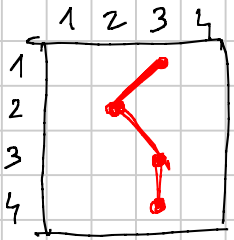
  S[i,j] + P[i,j]

}



aggiunta per ricostruire il cammino

```
Cerca casella (n);
max = S[1,1]; jmax = 1;
for j = 2 to n {
    if S[1,j] > max {
        max = S[1,j]
        jmax = j
    }
} return jmax;
```



M

			-1
1			
			0

max in S(1,3)