01-12-2022

CONVOLUZIONE

- L'operazione di **convoluzione** si fa fra **2 funzioni**;
- Nel caso discreto si può applicare solo quando ci sono determinate condizioni.

Per capire il concetto facciamo un esempio con un vettore:

Dato un vettore di lunghezza N, questo può essere pensato come un elemento di uno spazio N dimensionale.

234	204	34	16	44	134	12	11	56

Quindi possiamo scomporlo usando la base canonica d tale spazio.

- Qual è la base canonica?
- La sua base canonica è:

	234	204	34	16	44	134	12	11	56	=
234 *	1	0	0	0	0	0	0	0	0	+
204 *	0	1	0	0	0	0	0	0	0	+
34 *	0	0	1	0	0	0	0	0	0	+
16 *	0	0	0	1	0	0	0	0	0	+
44 *	0	0	0	0	1	0	0	0	0	+
134 *	0	0	0	0	0	1	0	0	0	+
12 *	0	0	0	0	0	0	1	0	0	+
11 *	0	0	0	0	0	0	0	1	0	+
56 *	0	0	0	0	0	0	0	0	1	

- Lo stesso ragionamento vale per le immagini

$$\begin{bmatrix} 234 & 204 & 34 \\ 16 & 44 & 134 \\ 12 & 11 & 56 \end{bmatrix} =$$

$$234 * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + 204 * \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + 34 * \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$16 * \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + 44 * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + 134 * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$12 * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} + 11 * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + 56 * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- In questo modo posso applicare un operatore locale, cioè si può scomporre un vettore e scriverlo in base funzione della base canonica.

OPERATORI LOCALI

- Il valore d'uscita di ogni pixel dipenda da un limitato intorno del corrispondente punto di input
- Sono usati per migliorare la qualità delle immagini o per estrarre delle informazioni dall'immagine
- Si possono pensare come filtraggi dell'immagine
- Un *filtraggio* è ottenuto facendo la **convoluzione tra l'immagine ed una matrice**.

Operatori lineari

Un operatore $F : V \to W$ si dice **LINEARE** se per ogni coppia di vettori v_1 e v_2 in V e per ogni coppia di scalari (a, b) si ha che:

$$F(a\, v_1 + b\, v_2) = a\, F(v_1) + b\, F(v_2)$$

La **CONSEGUENZA** è: se conosco **una base** di V ed il **comportamento dell'operatore su** F su ogni elemento di tale base, posso calcolare il comportamento di F su ogni elemento di V
.

In altre parole:

- lineare vuol dire che se ho 1 legge da $V \to W$ tra vettori
  - se F lineare, comunque io prendo una coppia (v_1, v_2) e per ogni coppia possibile di scalari si ha la linearità della somma e del prodotto.

Esercizio esame

- Esercizio con operatore **lineare**:

Dato l'operatore:

$F(x,y) = \left(\frac{x}{2}, \frac{y}{3}\right)$ è LINEARE?

PER ESSERE LINEARE DEVE VALERE:

$$\boxed{F(ax_1+bx_2, ay_1+by_2) = aF(x_1,y_1)+bF(x_2,y_2)}$$

VERIFICO USANDO LA LEGGE DI F:

$$\boxed{\left(\frac{ax_1+bx_2}{2}, \frac{ay_1+by_2}{3}\right)} \overset{?}{=} a\left(\frac{x_1}{2},\frac{y_1}{3}\right)+b\left(\frac{x_2}{2},\frac{y_2}{3}\right)=$$

x y

$$=\left(\frac{ax_1}{2},\frac{ay_1}{3}\right)+\left(\frac{bx_2}{2},\frac{by_2}{3}\right)=$$

Sommo membro a membro:

$$=\boxed{\left(\frac{ax_1+bx_2}{2}, \frac{ay_1+by_2}{3}\right)}$$

Quindi l'operatore è lineare!

- Esempio con operatore **non lineare**:

Dato l'operatore:

$F(x,y) = (255-x, 255-y)$ è LINEARE?

PER ESSERE LINEARE DEVE VALERE:

$$\boxed{F(ax_1+bx_2, ax_2+by_2) = aF(x_1,y_1)+bF(x_2,y_2)}$$

VERIFICO USANDO LA LEGGE DI F:

$$\boxed{(255-(ax_1+bx_2)),(255-(ay_1+by_2))} \overset{?}{=} a(255-x_1, 255-y_1)+b(255-x_2, 255-y_2)$$

$$(a\cdot 255-ax_1, a255-ay_1)+(b255-bx_2, b255-by_2)=$$

$$=\boxed{(a255+b255-ax_1-bx_2, a255-ay_1-by_2)}$$

Le due parti evidenziate in arancione
sono diverse fra loro. Pertanto l'operatore
che è stato proposto prima NON è LINEARE

INFATTI:

$255(a+b) \overset{?}{=} 255$ NO!

Operatori invarianti per traslazione

Se il **comportamento** dell'operatore **NON DIPENDE DALLA POSIZIONE** in cui io lo applico ma **DIPENDE SOLO DAI VALORI** che trovo in una posizione, allora tale operatore è **INVARIANTE PER TRASLAZIONE**.

Un esempio è possibile farlo con un operatore puntuale:

1. si considera un pixel;
2. si calcola il valore in base all'operatore puntuale scelto;
3. si ottiene il risultato.

Gli operatori visti fino ad ora *[Negativo, Incupimento, Schiarimento Logaritmico, Potenza Binarizzazione, Aumento/Diminuzione del contrasto, Solarizzazione]* sono invarianti per traslazione perchè non dipendono dalla posizione del pixel ma dal suo valore effettivo. (ovvero **NON** c'è **dipendenza dalle coordinate spaziali**)

Esempio di operatore non invariante per traslazione:

- "**BLURRING**", cioè l'applicazione dell'**effetto profondità di campo**. Questo operatore altera i pixel in base a dove essi si trovano. Se si ha una foto che ritrae una persona, la persona avrà 0 blur, mentre oltre i bordi del soggetto applica un certo blur in una determinata percentuale.

> In altre parole: se si ha una funzione $F(x, y)$ e se x e y sono **coordinate**, allora non si tratta di invariante per traslazione perchè x e y sono posizioni.

Definizione formale

Un operatore si dice **invariante per traslazione** (*shift invariant*) quando il suo comportamento sulle **immagini impulsive** è sempre il medesimo indipendentemente dalla posizione in cui si trova il pixel.

- Per *immagini impulsive* si intende quell'immagine dove c'è un solo pixel a 1 (*acceso*) e tutti gli altri a 0.
- Ovvero c'è solo un punto che ha il massimo valore e tutti gli altri hanno valore 0. Il risultato è uguale ovunque sia il valore massimo, appunto
- Infatti la **base canonica** è un **insieme di immagini impulsive.**

> *Come si è visto prima, tutti gli operatori puntuali sono invarianti per traslazione (anche se non sono lineari).*

Riassumendo:

Se F **è lineare** per descriverlo basta conoscere il comportamento su tutte le immagini impulsive

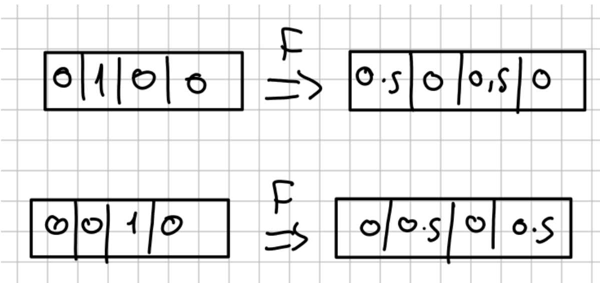
Se F **è shift invariant** si comporta allo stesso modo su tutti gli impulsi, indipendentemente dalla loro posizione

Se F è sia **lineare** che **shift invariant** per descriverlo basta conoscere come si comporta su **un solo** impulso.

La "**risposta all'impulso**" o "**point spread function**" di F è la carta di indentità di tale operatore.

La **linearità** è data da:

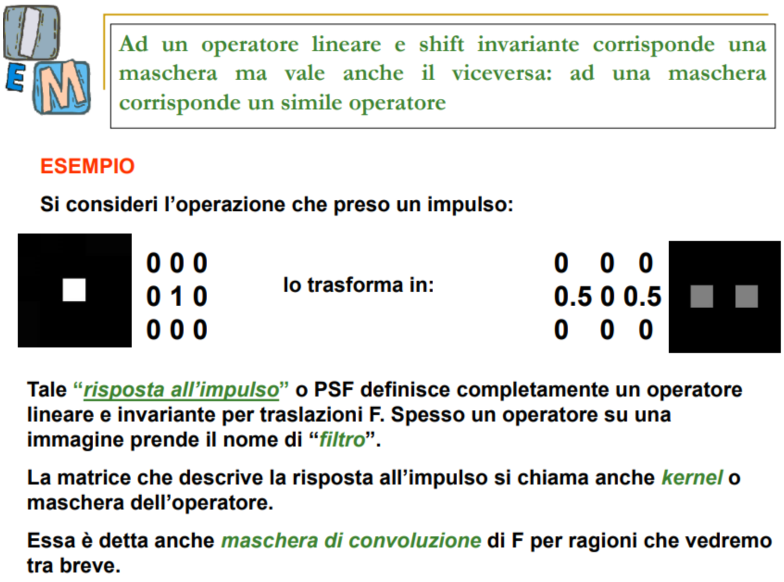
$$F(a\, v_1 + b\, v_1) = a\; F(v_1) + b\; F(v_2)$$

- Se v_1 è della base canonica allora il risultato per ogni v è uguale
- 

- Infatti, per esempio, *basta conoscere solo* $F(v_1)$. Se lo conosco, *posso applicare la stessa funzione semplicemente shiftandola*, come nell'esempio del disegno soprastante.

> *In altre parole*: **basta conoscere il risultato dell'applicazione ad un impulso solo**, ovvero ciò che lo caratterizza in maniera totale, ovvero si parla di **KERNEL**, attraverso il quale si può applicare un operatore locale *per convoluzione* e tale operatore deve essere kernel (cioè invariante per traslazione e lineare).

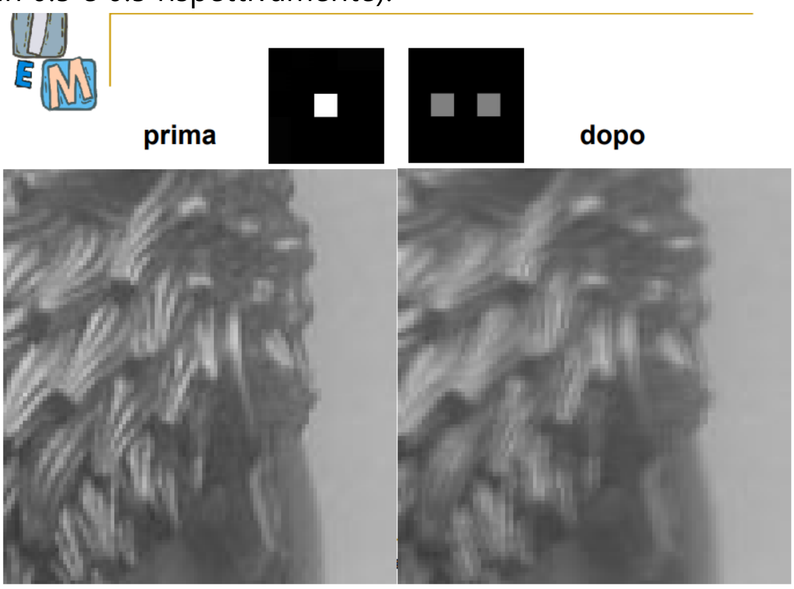
Esempio 1:



- *Praticamente*: se l'*operatore è lineare e invariante per traslazione*, si è in presenza di **KERNEL DELL'OPERATORE** dell'operatore o **MASCHERA DI CONVOLUZIONE**

Esempio 2:

Se applico l'operatore che fa *questa cosa* (cioè se c'è un pixel a 1, lo divide orizzontalmente in 0.5 e 0.5 rispettivamente):



Si è usata la definizione di funzione lineare senza usare la convoluzione:

- se fosse *400x600* avrei *24000* matrici (*impulsive*) e ognuna di esse sarà fatta da un 1 e tutte le altre posizioni uguali a 0;
- per ognuna delle posizioni dove trovo 1 e lo faccio diventare 0.5;
- e poi sommo insieme le matrici (*impulsive*) per ottenere l'immagine finale;
- la convoluzione fa esattamente questa cosa.

Esempio con i numeri:

Esempio senza convoluzione:

$$\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} \rightrightarrows \begin{bmatrix}0&0&0\\\frac{1}{2}&0&\frac{1}{2}\\0&0&0\end{bmatrix}$$

Prende il nome di KERNEL

Applico il KERNEL alla matrice:

$$F\left(\begin{bmatrix}2&4&6\\8&10&12\\14&16&18\end{bmatrix}\right)$$

La base canonica di F è:

$$2\cdot\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}+4\cdot\begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix}+\dots+10\cdot\begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix}+\dots=$$

Applico l'operatore:

$$2\cdot\begin{pmatrix}0&\frac{1}{2}&0\\0&0&0\\0&0&0\end{pmatrix}+4\cdot\begin{pmatrix}\frac{1}{2}&0&\frac{1}{2}\\0&0&0\\0&0&0\end{pmatrix}+\dots+10\cdot\begin{pmatrix}0&0&0\\\frac{1}{2}&0&\frac{1}{2}\\0&0&0\end{pmatrix}+\dots=\dots$$

$$\begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix}+\begin{pmatrix}2&0&2\\0&0&0\\0&0&0\end{pmatrix}+\dots+\begin{pmatrix}0&0&0\\5&0&5\\0&0&0\end{pmatrix}+\dots=$$

Se sommo tutti i valori di ogni singola matrice allora posso trovare la matrice con l'operatore F applicato alla matrice originale

- La **grandezza del kernel** può variare fino ad essere infinitamente grande
- Per **ragioni pratiche**, però, **si usano solo kernel con dimensioni finite** (perchè si userà con un processore).
- Possono essere molto grandi.
- Tendenzialmente non superano certe soglie 3×3, 5×5, 7×7 per piccole operazioni.

> Infatti, per operatori locali che coinvolgono più pixel (operatori più complessi, del tipo $15x15$, $21x21$, ecc..) le dimensioni del kernel sono più grandi.

Le dimensioni del kernel influenzano la complessità dell'operazione di filtraggio.
Tale *complessità* dipende ovviamente anche dal numero dei pixel di una immagine.

Più è grande il kernel $\implies$ più è il tempo richiesto per applicare tale operatore $\implies$ più il filtro che richiede tale kernel richiede tempo per essere applicato.

CONVOLUZIONE APPROFONDITA

Perché filtri "convolutivi"?

I filtri lineari e invarianti per traslazione vengono chiamati anche filtri convolutivi.

Dobbiamo studiare la **operazione di convoluzione** per capire meglio come un filtro può essere calcolato.

Inoltre la convoluzione è un fenomeno estremamente importante per ogni tipo di signal processing e per la descrizione di numerosi eventi fisici.

Convoluzione: proprietà

- Per indicare l'operazione di convoluzione si usa la notazione

$$h = f \otimes g$$

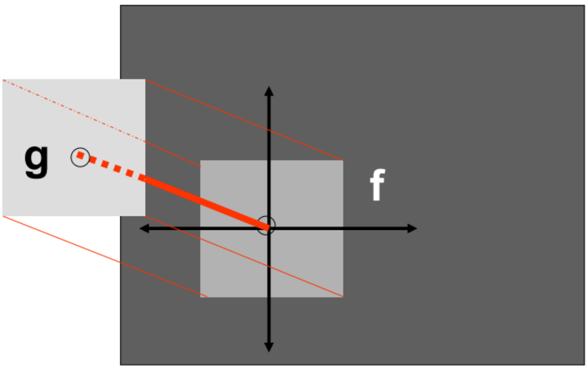
- La convoluzione è commutativa

$$f \otimes g = g \otimes f$$

- La convoluzione è associativa

$$(f \otimes g) \otimes h = f \otimes (g \otimes h)$$

La convoluzione applicata fra 2 matrici (*raster*) applico operatore locale a f la cui risposta impulsiva è proprio uguale a g:



Solitamente il kernel ha dimensioni più piccole dell''immagine.
In pratica:

1. per ogni posizione dell'immagine, si sovrappone il kernel
2. per ogni pixel si fa la combinazione fra pixel del kernel e il pixel effettivo
3. Ma come si mettono insieme?

Nel caso finito (1)

Se il kernel f ha dimensioni $k \times h$ la formula va riscritta nella seguente maniera:

$$h_{m,n} = \sum_{i=-\lfloor k/2 \rfloor}^{\lceil k/2 \rceil -1} \sum_{j=-\lfloor h/2 \rfloor}^{\lceil h/2 \rceil -1} \left(f_{i,j} * g_{m+i,n+j} \right)$$

	-1	0	1
-1	a	b	c
0	d	e	f
1	g	h	i

- Questa formula descrive il fatto che dopo aver sovrapposto il kernel all'immagine originale, per ogni punto di sovrapposizione si fa la somma dei prodotti punto a punto. Il numero risultante lo metto nella posizione corrispondente.
- In questo caso gli indici del kernel sono disposti in modo da avere il punto di coordinate $(0, 0)$ nella posizione centrale.

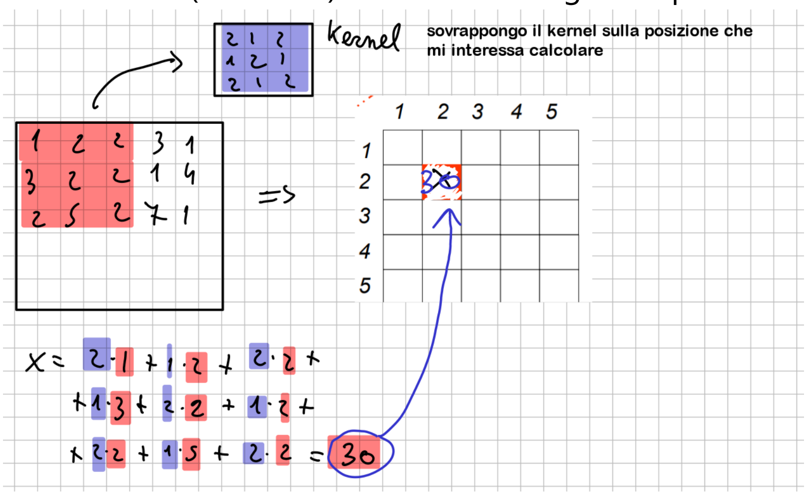
Nel caso finito (2)

Se il kernel f ha dimensioni $k \times h$ la formula va riscritta nella seguente maniera:

$$h_{m,n} = \sum_{i=1,j=1}^{k,h} f_{i,j} * g_{m+(i-k+\lfloor k/2 \rfloor),n+(j-h+\lfloor h/2 \rfloor)}$$

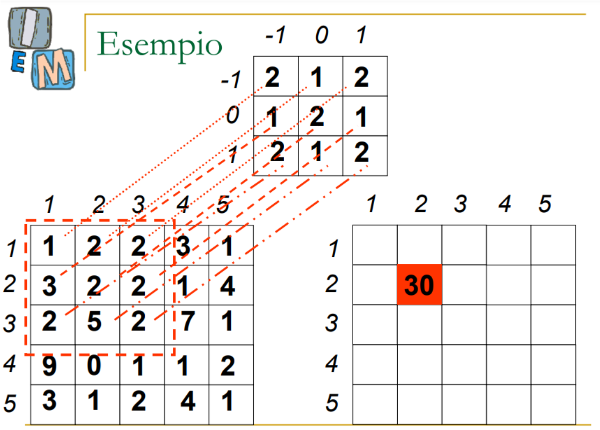
	1	2	3
1	a	b	c
2	d	e	f
3	g	h	i

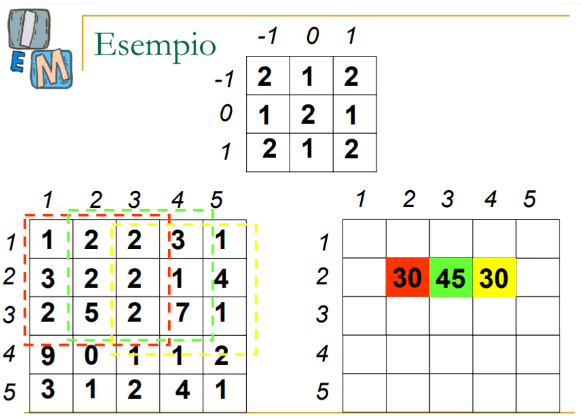
Le 2 formule (nei 2 casi) descrivono il seguente processo:

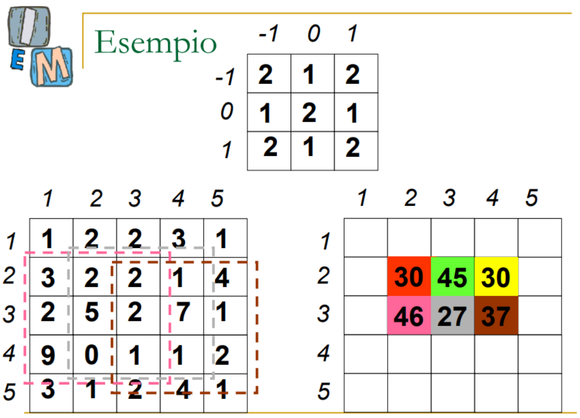


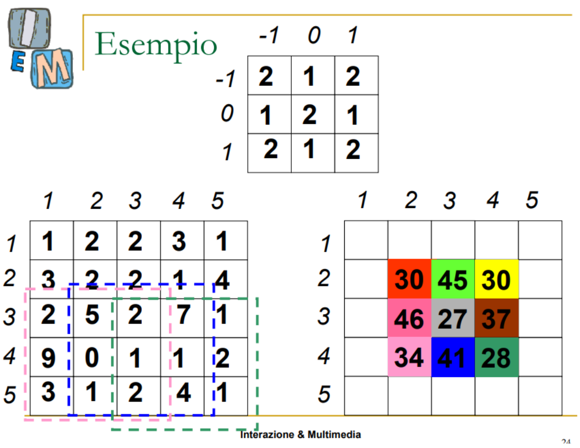
- Data una matrice (quella con l'area rossa selezionata) si decide dove si vuole applicare il kernel (matrice evidenziata di blu)
- si sovrappone il kernel alla posizione (2,2) [indici che partono da 1] della matrice iniziale, quindi con centro x
- Per ogni posizione della matrice iniziale (rosso) si moltiplica il valore corrispettivo nella stessa posizione del kernel (blu)

- Infine si sommano i valori ottenuti.









> Quindi **applicare un filtro lineare e shift invariante** ad una immagine è **equivalente** a calcolare la **convoluzione del kernel del filtro con l'immagine**.

Nell'implementazione

- Un problema è quello dei bordi: come fare la convoluzione e il filtraggio ai bordi? POSSIBILI SOLUZIONI:

1. **Filtrare solo le zone centrali dell'immagine** e scartare i bordi (le dimensioni diminuiscono);

    - Le aree in grigio non verranno calcolate

      input

1	2	2	3	1
3	2	2	1	4
2	5	2	7	1
9	0	1	1	2
3	1	2	4	1

      output

	30	45	30	
	46	27	37	
	34	41	28	

2. **Supporre che tutto intorno all'immagine ci sia 0**;
    - far finta che l'immagine sia *più grande* e abbia un bordo esterno tutto riempito di 0
    - input

0	0	0	0	0	0	0
0	1	2	2	3	1	0
0	3	2	2	1	4	0
0	2	5	2	7	1	0
0	9	0	1	1	2	0
0	3	1	2	4	1	0
0	0	0	0	0	0	0

      output

11	19	17	22	11
25	30	45	30	31
25	46	27	37	19
35	34	41	28	29
16	27	12	18	10

3. **Assumere una topologia "toroidale"**: quando si "sfora a destra" si rientra a sinistra, quando si "*sfora*" in basso di rientra in alto e viceversa;

    - input

1	3	1	2	4	1	3
1	1	2	2	3	1	1
4	3	2	2	1	4	3
1	2	5	2	7	1	2
2	9	0	1	1	2	9
1	3	1	2	4	1	3
1	1	2	2	3	1	1

      output

27	30	29	32	33
33	30	45	30	40
38	46	27	37	45
41	34	41	28	48
28	35	24	27	40

4. **Aggiungere delle righe/colonne** copiando i valori che si trovano immediatamente adiacenti alla riga/colonna corrispettiva.

    - input

1	1	2	2	3	1	1
1	1	2	2	3	1	1
3	3	2	2	1	4	4
2	2	5	2	7	1	1
9	9	0	1	1	2	2
3	3	1	2	4	1	1
3	3	1	2	4	1	1

      output

25	27	29	31	33
34	30	45	30	39
51	46	27	37	32
54	34	41	28	35
48	32	24	34	26