Aritmetica Modulare

L'idea di base dell'aritmetica modulare consiste nel definire le operazioni usando solo un set finito di elementi, come ad esempio i primi n interi $Z_n=\{0,...,n-1\}$ cercando di mantenere valide proprietà come commutatività, associatività, etc..

Somma

Dato il set dei primi n interi, $Z_n=\{0,...,n-1\}$, è possibile definire la somma su Z_n:

$$+_n: Z_n \times Z_n \rightarrow Z_n \quad , \quad a +_n b = \mathbf{(a + b) \bmod n}$$

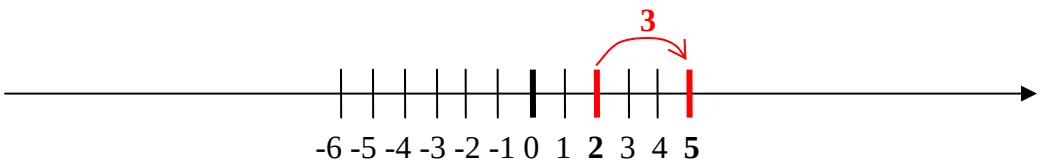
cioè dati due interi in Z_n **a** e **b**, la somma $+_n$ è un numero ancora in Z_n ed è uguale alla somma classica *modulo n*.

Ex: *Dato l'insieme $Z_4=\{0,1,2,3\}$, la somma modulo 4 ha la seguente struttura:*

$+_4$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Possiamo pensare gli **Z** come una successione di punti su una retta orientata infinita da ambo i lati. In questa immagine la somma in **Z** consiste nel muoversi dal primo termine nel verso dei positivi di una distanza uguale al secondo termine.

Ex: *Consideriamo la somma 2 + 3 = 5 in **Z**:*

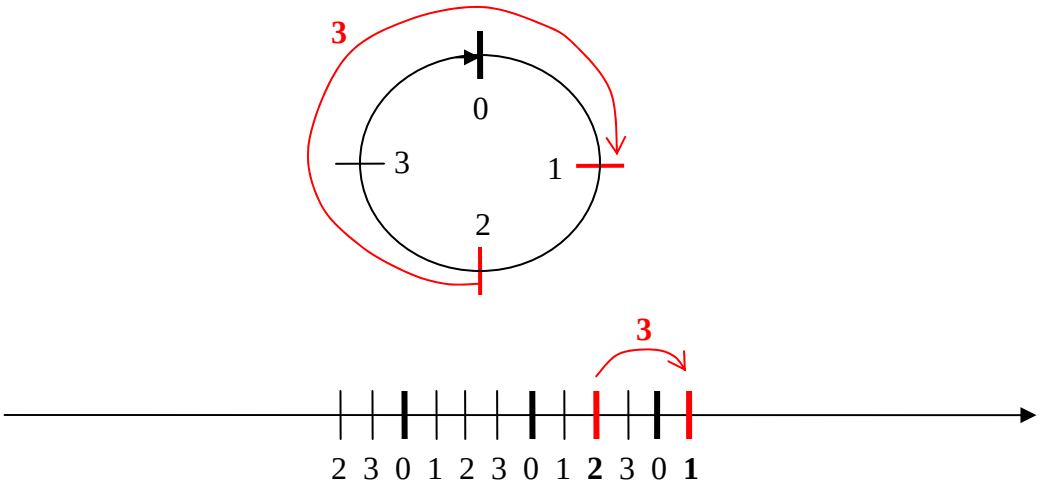


Analogamente possiamo pensare agli $\mathbf{Z_n}$ come un insieme di punti posti su di una circonferenza. Anche in questo caso la somma corrisponde ad uno

Massimo Marchi – marchi@dsi.unimi.it

spostamento nel verso dei positivi con la differenza che si *ritorna indietro nei negativi* quando si esaurisce la capacità espressiva degli Z_n.

Ex: *Consideriamo la somma* $2 +_4 3 = 1$ *in* $\mathbf{Z_4}$*:*



Rappresentante canonico

Dato un qualsiasi intero **i** in **Z**, segue che **i** mod **n** è un elemento di $\mathbf{Z_n}$. In altre parole il modulo **n** divide **Z** in **n** sottinsiemi, o *classi*, di cui possiamo pensare gli elementi di $\mathbf{Z_n}$ come dei *rappresentanti*.

Ex: *Dato l'insieme* $Z_4=\{0,1,2,3\}$*:*

nella classe di 0 stanno: 0, 4, -4, 8, -8, 12 ...
nella classe di 1 stanno: 1, 5, -3, 9, -7, 13...
nella classe di 2 stanno: 2, 6, -2, 10, -6, 14...
nella classe di 3 stanno: 3, 7, -1, 11, -5, 15...

Normalmente per indicare gli elementi di $\mathbf{Z_n}$ si usano i *rappresentanti canonici* {0, 1, 2,...., n-1} ma nulla vieta di utilizzare all'occorrenza altri rappresentanti, come {0, 1, 2,...., n/2-1, -n/2, -n/2+1,.....,-1}.

Ex: $\mathbf{Z_4} = \{0,1,2,3\} = \{0,1,-2-1\}$ *poiché:*

2 e -2 appartengono alla stessa classe: -2 mod 4 = 2
3 e -1 appartengono alla stessa classe: -1 mod 4 = 3

Opposto

L'opposto di un numero **a** è quel numero **b** che sommato ad **a** da come risultato **0**. Negli interi **Z** l'opposto di **a** è banalmente **–a**.

Ex: *Dato l'insieme Z_4={0,1,2,3}:*

*L'opposto di **0** è **0** : $0 +_4 0 = (0 + 0) \bmod 4 = 0 \bmod 4 = 0$*
*L' opposto di **1** è **3** : $1 +_4 3 = (1 + 3) \bmod 4 = 4 \bmod 4 = 0$*
*L' opposto di **2** è **2** : $0 +_4 0 = (2 + 2) \bmod 4 = 4 \bmod 4 = 0$*
*L' opposto di **3** è **1** : $0 +_4 0 = (3 + 1) \bmod 4 = 4 \bmod 4 = 0$*

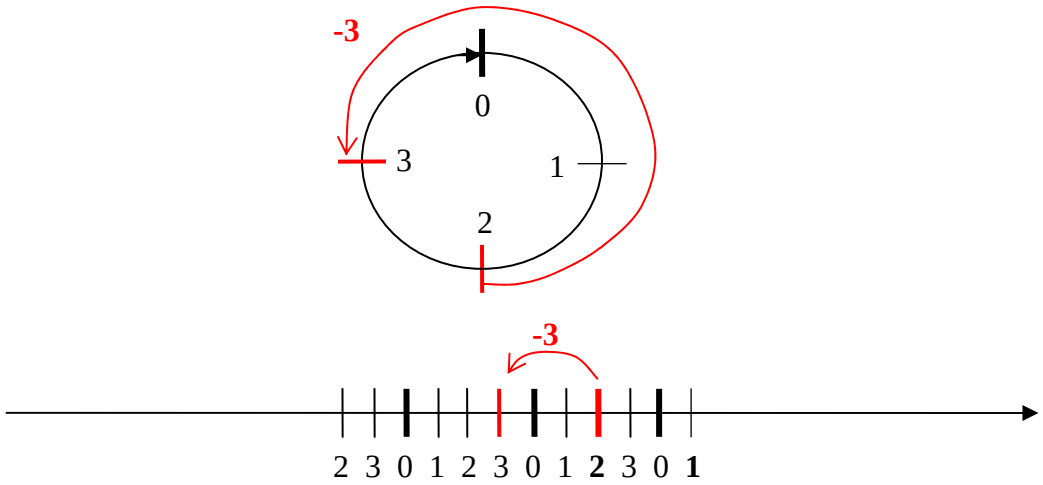
Sottrazione

Come per il caso naturale, possiamo definire la sottrazione in Z_n:

$$-_n : Z_n \times Z_n \rightarrow Z_n \quad , \quad a -_n b = a +_n (-b)$$

cioè la sottrazione si esegue sommando al primo termine l'opposto del secondo.

Ex: *Consideriamo la sottrazione $2 -_4 3 = 2 +_4 1 = 3$ in Z_4:*



Si dimostra che:

$$a -_n b = a +_n (-b) = (a + (-b)) \bmod n = \mathbf{(a - b) \bmod n}$$

cioè, è possibile, come per la somma, eseguire i calcoli in Z ed applicare il modulo solo alla fine, per ritornare al rappresentante canonico.

Massimo Marchi – marchi@dsi.unimi.it

Aritmetica modulare a n bit

Calcolare una somma di parole di n bit utilizzando un sommatore a n bit equivale ad eseguire i calcoli in aritmetica modulo 2^n:

Ex: *un sommatore a 8 bit implementa un aritmetica modulo 256.*

Utilizzare la notazione in **complemento a due** per i negativi equivale ad implementare un aritmetica modulo 2^n prendendo come rappresentanti l'insieme da $\mathbf{-2^n/2}$ a $\mathbf{2^n/2-1}$. Infatti il complemento a due di un numero non è altro che il primo rappresentante negativo del suo opposto.

Ex: *Dati 8 bit, il CA2 di -2 è:*

$$-1 = not(00000010) +_{255} 1 = 11111101_2 +_{255} 1_2 = 11111110_2 = 254$$

ma 254 è l'opposto canonico di 2, infatti:

$$2 + 254 = 256 \bmod 256 = 0$$

Da notare che eseguire il **not** di una parola di n bit equivale a sommare modulo 2^n l'*offset* 2^n-1

Ex: *Dati 4 bit:*

$$(2^4-1) +_{16} 2 = (10000_2 - 1_2) +_{16} 2 = 1111_2 +_{16} 0010_2 = 1101_2$$

da cui calcolare il CA2 equivale a sommare 2^n

Ex: *Dati 4 bit:*

$$-2 = 0 +_{256} (-2) = (0 - 2) \bmod 256 = (0 - 2 + 256) \bmod 256.$$

Analogamente si dimostra che utilizzare il **CA1** per eseguire le sottrazioni equivale ad implementare un aritmetica modulo (2^n-1)

Massimo Marchi – marchi@dsi.unimi.it