

Ricordiamo che il problema era il seguente:

$$\text{SUM} = \max_{(i,j)} \left\{ \sum_{k=i}^j a_k, 0 \right\} \text{ con complessità dell'ordine di } m^3$$

$1 \leq i \leq j \leq m$

L'algoritmo calcola il valore della coppia generata  $(i,j)$  e controlla che sia il massimo o meno.

Si puo' notare perci' che per  $j > i$  i valori sono cosi' calcolati:

$$\sum_{z=i}^j a_z = a_j + \sum_{z=i}^{j-1} a_z$$

$\underbrace{\phantom{\sum_{z=i}^j a_z}}_{\text{SUM}(i,j)}$        $\underbrace{\phantom{\sum_{z=i}^{j-1} a_z}}_{\text{j-esimo elemento}}$

$+ \text{SUM}(i, j-1)$

Quindi l'algoritmo ricalcola inutilmente una sottosequenza aggiungendo solo un elemento. Fatta queste considerazione l'algoritmo potrebbe essere:

ALG ( $A, m$ )

MAXSUM=0

FOR  $i=1$  TO  $m$  DO

genera posizione iniziale

SUM=0

FOR  $j=i$  TO  $m$  DO

genera posizione finale

SUM +=  $A[j]$

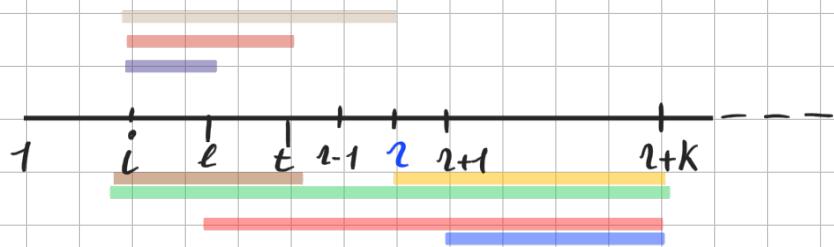
la sottosequenza (SUM) viene conservata e gli si aggiunge il  $j$ -esimo elemento

$$\text{MAX SUM} = \text{MAX}(\text{MAXSUM}, \text{SUM})$$

RETURN MAX SUM

Analogamente a studiare il numero di operazioni di base eseguite abbiamo ridotto sensibilmente la complessità (ora dell'ordine di  $n^2$ )

A questo punto migliorare l'algoritmo diventa praticamente impossibile perché solamente per generare le coppie abbiamo una complessità di  $n^2$  quindi l'unico modo che abbiamo per ridurre questo numero dobbiamo evitare di generare tutte le coppie (e quindi le sottosequenze). Ma quali sottosequenze andiamo a scartare?



Data la nostra sequenza sequenza individuiamo l' $i$ -esimo elemento con  $i | V_i \leq t \leq 2-1 | \sum_{z=i}^t a_z \geq 0$  quindi le sottosequenze degli elementi da  $i$  a  $2-1$  sono **TUTTE POSITIVE**. Posta quest'ipotesi vanno le seguenti proprietà:

$$\textcircled{1} \sum_{z=i}^l a_z \leq \sum_{z=i}^t a_z \quad (\text{ogni sottosequenza di elementi positivi sarà crescente})$$

$$\textcircled{2} \sum_{z=i}^1 a_z \geq 0 \Rightarrow \sum_{z=i}^{2+k} a_z \leq \sum_{z=1}^{2+k} a_z \quad \forall i \leq l \leq r \quad \forall k \geq 1$$

Fondamentalmente il discorso è analogo al punto precedente, se da  $i$  a  $r$  è crescente allora "sommando" questa quantità a qualsiasi altro intervallo ottengono inevitabilmente sottosequenze dal valore maggiore.

$$\textcircled{3} \sum_{z=i}^r a_z < 0 \Rightarrow \sum_{z=l}^{2+k} a_z < \sum_{z=r+1}^{2+k} a_z \quad \forall i \leq l \leq r \quad \forall k \geq 1$$

Sappiamo che fino a  $l \leq r-1$  la sequenza è positiva. Se considerato  $r$  diventa negativa allora  $r$  è un numero negativo minore di tutto la sequenza che lo precede.

Ma da se quindi che le sequenze che vanno da  $r+1$  a  $2+k$  saranno maggiori di quelle da  $l$  ( $< r$ ) a  $r+k$  perché  $r$  "annulla" tutto il precedente

OSS

② e ③ sono così distinti complementari

Tutto ciò comporta che dovremo scegliere sicuramente una sottosequenza che parte da  $i+1$ . Vediamo l'algoritmo

ALGO 2(A, N)

MAXSUM = 0

SUM = 0

FOR  $J=1$  TO  $N$  DO (praticamente  $i=1$ )

SUM += A[J]

IF SUM < 0 THEN  $J=2$  del nostro caso (caso ③)

SUM = 0

Così settiamo l'inizio a  $i+1$

ELSE

MAXSUM += A[J]

RETURN MAXSUM

La complessità di quest'algoritmo è  $\Theta(N)$  (1 ciclo for)

## STRUTTURE DI CONTROLLO

Le strutture di controllo influenzano fortemente la complessità di un algoritmo. Il ciclo for esplicita quante ripetizioni verranno effettuate, il ciclo while e la ricorsione no, bisogna fare dei ragionamenti sui vari casi.

Alcuni algoritmi possono essere migliori sul breve termine oppure per valori di input bassi.

## ORDINAMENTO

Possiamo vedere l'**ORDINAMENTO** di una sequenza  $A = a_1, \dots, a_n$

come una **PERMUTAZIONE**

$\pi = a_{i_1}, \dots, a_{i_m} \mid \forall i \leq j \leq m \ a_{i_j} \in A \wedge a_{i_j} \in \pi \text{ e in particolare}$   
 $a_{i_j} \leq a_{i_{j+1}}$  (deve essere ordinata)

Se l'ordinamento è una permutazione ordinata potremo pensare di generare tutte le permutazioni di una sequenza  $A$  di lunghezza  $n$  finché non trovo quella ordinata.

Esistono  $n!$  permutazioni per  $A$ , dell'ordine di  $n^n$  (molto complesso)