# Algoritmo di Kadane

Alessandro Ferro

22 aprile 2022

**Sommario**

Si vuole dimostrare la correttezza dell'Algoritmo di Kadane, in particolare la versione che ammette l'esistenza di una sotto-sequenza vuota (ovvero l'algoritmo non restituirà mai un valore inferiore a 0).

## 1 Teorema

Per dimostrare la correttezza dell'algoritmo, enunciamo il seguente teorema:
Fissati $i, r \in \mathbb{N}$, se $\forall j$ compreso nell'intervallo $i \leq j < r$ si verifica che

- $SUM(i, j) \geq 0$
- $SUM(i, r) < 0$

$\implies$

1. $SUM(p, q) \leq SUM(i, q)$ $\forall p, q$ comprese nell'intervallo $i \leq p \leq q \leq r$
2. $SUM(p, r + k) < SUM(r + 1, r + k)$ $\forall p$ comprese nell'intervallo $i \leq p \leq r$ con $k \geq 1$

### 1.1 Dimostrazione informale

Dimostriamo dapprima il punto 1:
Tutte le sequenze $(i, i), (i + 1), (i + 2), ..., (i, q)$ hanno somma non negativa per ipotesi (in quanto $q$ è nell'intervallo $i \leq q < r$), pertanto è evidente che una somma che ha più termini ($SUM(i, q)$) sia maggiore o uguale di una somma che ne ha di meno o in egual misura ($SUM(p, q)$).

Adesso dimostriamo il punto 2:
Tramite una più formale dimostrazione scopriremo che $SUM(p, r) < 0$. Avremo quindi che
$SUM(p, r + k) = SUM(p, r) + SUM(r + 1, r + k) \implies SUM(p, r + k) < SUM(r + 1, r + k)$.

### 1.2 Dimostrazione

Dimostrazione punto 1:
Se $p = i$ allora si ha banalmente $SUM(p, q) = SUM(i, q)$. Dimostriamo allora quando $i < p$.

$$SUM(i, q) = SUM(i, p - 1) + SUM(p, q)$$
$$\implies$$
$$SUM(p, q) = SUM(i, q) - SUM(i, p - 1)$$

Sappiamo che $SUM(i, p - 1) \geq 0$ in quanto $i \leq p - 1 < r$

Ma allora

$$SUM(p, q) \leq SUM(i, q)$$

Dimostrazione punto 2:

$$SUM(p, r) = SUM(p, r-1) + SUM(r, r)$$

Sappiamo che $SUM(p, r-1) \leq SUM(i, r-1)$ grazie al teorema del punto 1 supponendo $q = r-1$
Quindi

$$SUM(p, r-1) \leq SUM(i, r-1)$$
$$\Longrightarrow$$
$$SUM(p, r-1) + SUM(r, r) \leq SUM(i, r-1) + SUM(r, r)$$

Sapendo che $SUM(i, r-1) + SUM(r, r) = SUM(i, r)$ e sapendo inoltre che $SUM(i, r) < 0$ si ha che

$$SUM(p, r) \leq SUM(i, r) < 0$$
$$\Longrightarrow$$
$$SUM(p, r) < 0$$

Inoltre sappiamo che

$$SUM(p, r+k) = SUM(p, r) + SUM(r+1, r+k)$$
$$\Longrightarrow$$
$$SUM(p, r+k) < SUM(r+1, r+k)$$

In quanto $SUM(p, r) < 0$.

### 1.3 Significato e applicazione del teorema

Se sappiamo che si verifica questa proprietà allora

- Grazie al punto 1 possiamo dire che $SUM(x, y)$ con $i < x \leq y \leq r$, ovvero $SUM()$ delle sotto-sequenze che iniziano e finiscono nell' intervallo $(i, r]$ sarà sempre minore o uguale a $SUM(i, y)$. Il valore di $SUM(i, y)$ lo si conosce già, pertanto è inutile generare tali sotto-sequenze. In altre parole, sia data la sequenza $(i, y)$. Nessuna scelta di $x$ tale che $i < x \leq r$ può dare luogo a un valore di $SUM(x, y)$ maggiore di $SUM(i, y)$.
  Poiché il valore di $SUM(i, y)$ è stato scoperto già (in quanto abbiamo già analizzato tutte le sequenze $(i, i), (i, i+1), ..., (i, r)$), non è necessario generare la sequenza $(x, y)$.
  Se per assurdo si avesse che $SUM(x, y) > SUM(i, y)$, la sotto-sequenza localmente massima potrebbe essere $(x, y)$, e dunque sarebbe necessario valutarne il valore della somma.

  Comunque, da un altro punto di vista, il motivo per non generare sequenze inutili è osservare che poiché la sotto-sequenza $(i, r-1)$ contiene solo elementi non negativi, è chiaro che localmente la sotto-sequenza di somma massima è $(i, r-1)$.

- Grazie al punto 2 possiamo dire che è inutile analizzare le sequenze $(x, r+k)$ con $x$ nell'intervallo $[i, r]$ perché più avanti si troverà sicuramente una sottosequenza di somma maggiore.

Dunque, se abbiamo analizzato le sequenze $(i, i), (i, i+1), (i, i+2), ..., (i, r)$ sarà inutile analizzare una sequenza $(x, y)$ con $x$ compreso nell'intervallo $i < x \leq r$, perché comunque preso $y$, sarà inutile la sua analisi.

## 2 Algoritmo

```
KadaneAlgorithm(Array v){
        n = v.Length();
        max_sum = 0;
        local_sum = 0;

        for(i = 0 ; i < n ; i++){
                local_sum = local_sum + v[i];
                if( local_sum < 0 ){
                        local_sum = 0;
                }
                else if( local_sum > max_sum ){
                        max_sum = local_sum;
                }
        }

        return max_sum;
}
```

### 2.1 Spiegazione

L'algoritmo ha una complessità $T(n) = \Theta(n)$ in quanto non genera tutte le sottosequenze possibili (altrimenti il tempo sarebbe stato un $\Omega(n^2)$).

Infatti se trova che $SUM(i, r) < 0$ resetta *sum* a 0 e riparte iniziando a sommare dalla cella $r + 1$ in avanti, in virtù del fatto che generare qualsiasi sotto-sequenza $(x, y)$ con $i < x \leq r$ sarebbe inutile.