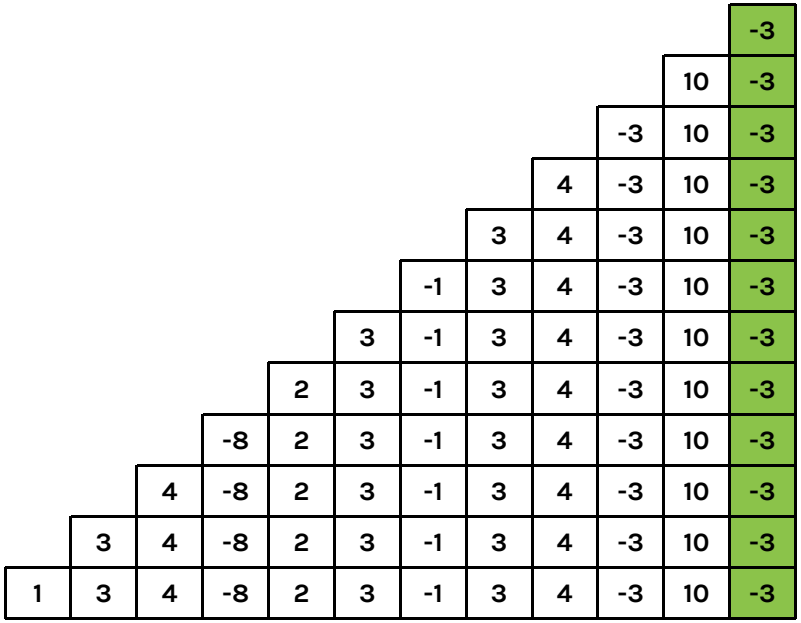
Sottovettore di somma massimale

1	3	4	-8	2	3	-1	3	4	-3	10	-3	2

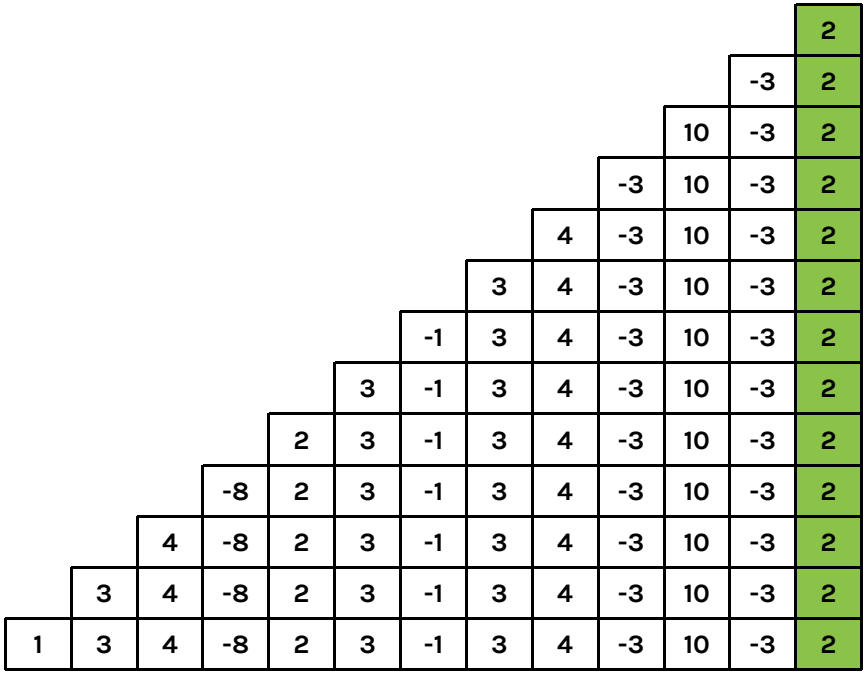
- Proviamo ad adottare un approccio *brute force*
- Calcoliamo la somma di ogni possibile sottovettore che termina in A[i]
- Dopo, calcoliamo la soma di ogni possibile sottovettore che termina in A[i+1]
- Il valore massimo tra tutte le somme identifica il nostro sottovettore di interesse

Sottovettore di somma massimale

1	3	4	-8	2	3	-1	3	4	-3	10	-3	2

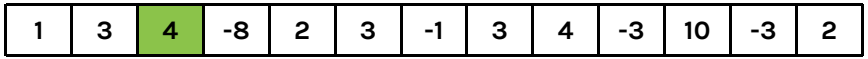


1	3	4	-8	2	3	-1	3	4	-3	10	-3	2

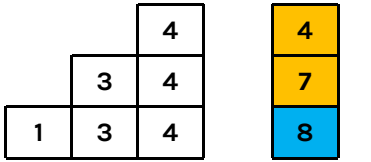


Sottovettore di somma massimale

- Concentriamoci sul terzo elemento del vettore

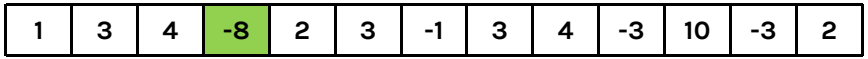


- Il “massimo locale” è 8, corrispondente al sottovettore dall'indice 0 all'indice 2

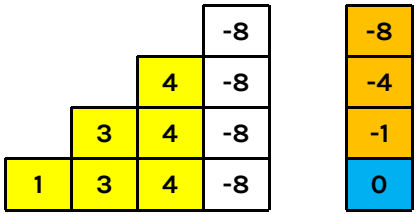


Sottovettore di somma massimale

- Concentriamoci sul quarto elemento del vettore



- La parte gialla corrisponde all'insieme di vettori considerato nel caso precedente
- Se conosciamo già quelle somme, non è necessario ricalcolarle
- Se ci confrontiamo con il massimo precedente e troviamo uno zero (o un numero negativo), comincia una nuova fetta di sottovettore



somme

Programmazione dinamica

Programmazione dinamica

- Sia *maxHere[i]* il valore del sottovettore di somma massima che termina in posizione *A[i]*

$$maxHere[i] = \begin{cases} 0 & i < 0 \\ \max(maxHere[i-1] + A[i], 0) & i \geq 0 \end{cases}$$

- Viene tenuta traccia di quanto calcolato fino ad un certo punto di esecuzione dell'algoritmo

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere													
maxSoFar													
last													
start													
end													

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere													
maxSoFar													
last													
start													
end													

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0      # Maximum found so far
    maxHere = 0       # Maximum slice ending at the current pos
    start = end = 0   # Start, end of the maximal slice found so far
    last = 0          # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	0												
maxSoFar	0												
last	0												
start	0												
end	0												

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=0

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	0												
maxSoFar	0												
last	0												
start	0												
end	0												

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=0

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1												
maxSoFar	0												
last	0												
start	0												
end	0												

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=0

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1												
maxSoFar	1												
last	0												
start	0												
end	0												

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=1

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	1											
maxSoFar	1	1											
last	0	0											
start	0	0											
end	0	0											

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=1

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4											
maxSoFar	1	1											
last	0	0											
start	0	0											
end	0	0											

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=1

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4											
maxSoFar	1	4											
last	0	0											
start	0	0											
end	0	1											

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=2

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	4										
maxSoFar	1	4	4										
last	0	0	0										
start	0	0	0										
end	0	1	1										

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=2

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8										
maxSoFar	1	4	1										
last	0	0	0										
start	0	0	0										
end	0	1	1										

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=2

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8										
maxSoFar	1	4	8										
last	0	0	0										
start	0	0	0										
end	0	1	2										

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=3

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	8									
maxSoFar	1	4	8	8									
last	0	0	0	0									
start	0	0	0	0									
end	0	1	2	2									

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=3

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0									
maxSoFar	1	4	8	8									
last	0	0	0	0									
start	0	0	0	0									
end	0	1	2	2									

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=3

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0									
maxSoFar	1	4	8	8									
last	0	0	0	4									
start	0	0	0	0									
end	0	1	2	2									

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=4

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	0								
maxSoFar	1	4	8	8	8								
last	0	0	0	4	4								
start	0	0	0	0	0								
end	0	1	2	2	2								

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=4

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2								
maxSoFar	1	4	8	8	8								
last	0	0	0	4	4								
start	0	0	0	0	0								
end	0	1	2	2	2								

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=5

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	2							
maxSoFar	1	4	8	8	8	8							
last	0	0	0	4	4	4							
start	0	0	0	0	0	0							
end	0	1	2	2	2	2							

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=5

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5							
maxSoFar	1	4	8	8	8	8							
last	0	0	0	4	4	4							
start	0	0	0	0	0	0							
end	0	1	2	2	2	2							

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=6

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	5						
maxSoFar	1	4	8	8	8	8	8						
last	0	0	0	4	4	4	4						
start	0	0	0	0	0	0	0						
end	0	1	2	2	2	2	2						

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=6

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4						
maxSoFar	1	4	8	8	8	8	8						
last	0	0	0	4	4	4	4						
start	0	0	0	0	0	0	0						
end	0	1	2	2	2	2	2						

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=7

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	4					
maxSoFar	1	4	8	8	8	8	8	8					
last	0	0	0	4	4	4	4	4					
start	0	0	0	0	0	0	0	0					
end	0	1	2	2	2	2	2	2					

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=7

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7					
maxSoFar	1	4	8	8	8	8	8	8					
last	0	0	0	4	4	4	4	4					
start	0	0	0	0	0	0	0	0					
end	0	1	2	2	2	2	2	2					

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=8

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	7				
maxSoFar	1	4	8	8	8	8	8	8	8				
last	0	0	0	4	4	4	4	4	4				
start	0	0	0	0	0	0	0	0	0				
end	0	1	2	2	2	2	2	2	2				

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=8

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11				
maxSoFar	1	4	8	8	8	8	8	8	8				
last	0	0	0	4	4	4	4	4	4				
start	0	0	0	0	0	0	0	0	0				
end	0	1	2	2	2	2	2	2	2				

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=8

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11				
maxSoFar	1	4	8	8	8	8	8	8	11				
last	0	0	0	4	4	4	4	4	4				
start	0	0	0	0	0	0	0	0	4				
end	0	1	2	2	2	2	2	2	8				

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=9

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	11			
maxSoFar	1	4	8	8	8	8	8	8	11	11			
last	0	0	0	4	4	4	4	4	4	4			
start	0	0	0	0	0	0	0	0	4	4			
end	0	1	2	2	2	2	2	2	8	8			

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=9

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8			
maxSoFar	1	4	8	8	8	8	8	8	11	11			
last	0	0	0	4	4	4	4	4	4	4			
start	0	0	0	0	0	0	0	0	4	4			
end	0	1	2	2	2	2	2	2	8	8			

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=10

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8	8		
maxSoFar	1	4	8	8	8	8	8	8	11	11	11		
last	0	0	0	4	4	4	4	4	4	4	4		
start	0	0	0	0	0	0	0	0	4	4	4		
end	0	1	2	2	2	2	2	2	8	8	8		

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=10

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8	18		
maxSoFar	1	4	8	8	8	8	8	8	11	11	11		
last	0	0	0	4	4	4	4	4	4	4	4		
start	0	0	0	0	0	0	0	0	4	4	4		
end	0	1	2	2	2	2	2	2	8	8	8		

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=10

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8	18		
maxSoFar	1	4	8	8	8	8	8	8	11	11	18		
last	0	0	0	4	4	4	4	4	4	4	4		
start	0	0	0	0	0	0	0	0	4	4	4		
end	0	1	2	2	2	2	2	2	8	8	10		

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=11

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8	18	18	
maxSoFar	1	4	8	8	8	8	8	8	11	11	18	18	
last	0	0	0	4	4	4	4	4	4	4	4	4	
start	0	0	0	0	0	0	0	0	4	4	4	4	
end	0	1	2	2	2	2	2	2	8	8	10	10	

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=11

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8	18	15	
maxSoFar	1	4	8	8	8	8	8	8	11	11	18	18	
last	0	0	0	4	4	4	4	4	4	4	4	4	
start	0	0	0	0	0	0	0	0	4	4	4	4	
end	0	1	2	2	2	2	2	2	8	8	10	10	

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

i=12

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8	18	15	15
maxSoFar	1	4	8	8	8	8	8	8	11	11	18	18	18
last	0	0	0	4	4	4	4	4	4	4	4	4	4
start	0	0	0	0	0	0	0	0	4	4	4	4	4
end	0	1	2	2	2	2	2	2	8	8	10	10	10

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

i=12

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8	18	15	17
maxSoFar	1	4	8	8	8	8	8	8	11	11	18	18	18
last	0	0	0	4	4	4	4	4	4	4	4	4	4
start	0	0	0	0	0	0	0	0	4	4	4	4	4
end	0	1	2	2	2	2	2	2	8	8	10	10	10

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
    maxSoFar = 0     # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0 # Start, end of the maximal slice found so far
    last = 0         # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8	18	15	17
maxSoFar	1	4	8	8	8	8	8	8	11	11	18	18	18
last	0	0	0	4	4	4	4	4	4	4	4	4	4
start	0	0	0	0	0	0	0	0	4	4	4	4	4
end	0	1	2	2	2	2	2	2	8	8	10	10	10

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8	18	15	17
maxSoFar	1	4	8	8	8	8	8	8	11	11	18	18	18
last	0	0	0	4	4	4	4	4	4	4	4	4	4
start	0	0	0	0	0	0	0	0	4	4	4	4	4
end	0	1	2	2	2	2	2	2	8	8	10	10	10

Programmazione dinamica (Kadane's Algorithm)

```
def maxsum4(A):
  maxSoFar = 0     # Maximum found so far
  maxHere = 0      # Maximum slice ending at the current pos
  start = end = 0 # Start, end of the maximal slice found so far
  last = 0         # Beginning of the maximal slice ending here
  for i in range(0, len(A)):
    maxHere = maxHere + A[i]
    if maxHere <= 0:
      maxHere = 0
      last = i+1
    if maxHere > maxSoFar:
      maxSoFar = maxHere
      start, end = last, i
  return (start, end)
```

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8	18	15	17
maxSoFar	1	4	8	8	8	8	8	8	11	11	18	18	18
last	0	0	0	4	4	4	4	4	4	4	4	4	4
start	0	0	0	0	0	0	0	0	4	4	4	4	4
end	0	1	2	2	2	2	2	2	8	8	10	10	10