Lezione 5

Sottoarray di somma massima

Rossano Venturini

rossano@di.unipi.it

Pagina web del corso

http://didawiki.cli.di.unipi.it/doku.php/informatica/all-b/start

Esercizio 1

My strcat 1

Esercizio

Implementare la funzione

```
char* my_strcat1(char *s1, char *s2)
```

che restituisce un puntatore alla *nuova* stringa ottenuta concatenando le stringhe puntate da s1 e s2.

Scrivere un programma che legga due stringhe da tastiera e stampi la stringa ottenuta concatenandole. Si può assumere che le stringhe in input contengano non più di 1000 caratteri.

Notare che il comportamento di `my_strcat1()` è diverso da quello della funzione `strcat()` presente nella libreria **string**.

L'input è formato da due stringhe di lunghezza non maggiore di 1000 caratteri.
L'unica riga dell'output contiene la stringa ottenuta concatenando nell'ordine le due stringhe inserite.

Esercizio 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* my_strcat(char *x, char *y) {
  int xlen = strlen(x);
  int ylen = strlen(y);

  char* str = malloc(sizeof(char)*(xlen + ylen + 1));
  if (str == NULL) exit(1);

  int pos = 0, i= 0;
  for (i = 0; i < xlen; i++) {
    str[pos++] = x[i];
  }
  for (i = 0; i < ylen; i++) {
    str[pos++] = y[i];
  }
  str[pos] = '\0';
  return str;
}
```

Esercizio 1

```
int main(void) {
  char *cat;
  char x[MAXLEN];
  char y[MAXLEN];
  scanf("%s", x);
  scanf("%s", y);
  cat = my_strcat(x, y);
  printf("%s\n",cat);
  return 0;
}
```

Esercizio 5

My strcpy

Esercizio

Scrivere una funzione

`char* my_strcpy(char* dest, char* src)`

che copi *src* in *dest* (incluso il terminatore `'\0'`) e restituisca un puntatore a *dest*. La funzione assume che in *dest* vi sia spazio sufficiente per contenere *src* (è compito del chiamante assicurarsi che ciò sia vero).
Si noti che il comportamento di `my_strcpy()` è uguale a quello della funzione `strcpy()` presente nella libreria **string**.

Scrivere poi un programma che: legga una stringa da tastiera (di lunghezza non maggiore di 1000 caratteri); allochi spazio sufficiente per una seconda stringa destinata a contenere la prima; copi la prima stringa nella seconda; stampi la seconda stringa.

L'input è formato da una sola riga contenente una stringa di lunghezza non maggiore di 1000 caratteri.
L'unica riga dell'output contiene la stampa della seconda stringa.

Esercizio 5

```
 #include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXLEN (1001)

char* my_strcpy(char* dest, char* src) {
  char *s = src, *d = dest;

  while ((*d = *s) != '\0') {
    d++;
    s++;
  }

  return dest;
}
```

Esercizio 5

```
int main(void) {
  char y[MAXLEN], *x;
  int len;

  scanf("%s", y);
  len = strlen(y);
  x = malloc(sizeof(char) * (len+1));
  x = my_strcpy(x, y);
  printf("%s\n", x);

  return 0;
}
```

Esercizio 6 Moltiplicazione di stringhe

Esercizio

Si scriva una funzione

`char* product(char *str, int k)`

che data una stringa *str* e un intero *k* restituisca una stringa ottenuta concatenando *k* volte la stringa *str*.

Si scriva un programma che legga in input:

- una stringa (assumendo che la stringa sia non più lunga di 1000 caratteri);
- un intero, che indica quante volte ripetere la stringa.

e infine stampi l'output di `product()`.

L'input è costituito, nell'ordine, da: una stringa di lunghezza non superiore a 1000 caratteri; un intero k che indica quante volte ripetere la stringa inserita.
L'unica riga dell'output è formata da una stringa contenente k concatenazioni della stringa data in input.

Esercizio 6

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAXSIZE (1001)

char* product(char *str, int k) {
  int i;
  int len = strlen(str);
  int plen = len*k+1;

  char *prod = malloc(plen*sizeof(char));

  for(i = 0; i < plen-1; i++) {
    prod[i]= str[ i%len ];
  }
  prod[plen-1] = '\0';

  return prod;
}
```

Esercizio 6

```
int main(void) {
  char str[MAXSIZE], *prod;
  int k;
  scanf("%s", str);
  scanf("%d", &k);
  prod = product(str, k);
  printf("%s\n", prod);
  return 0;
}
```

Sottoarray di Somma Massima

Sottoarray di Somma Massima

Problema: Dato un array a di n interi (positivi e negativi) individuare il sottoarray di somma massima.

Input: un array a di interi

Output: la somma del sottrarray di somma massima

Sottoarray di Somma Massima

Problema: Dato un array a di n interi (positivi e negativi) individuare il sottoarray di somma massima.

Input: un array a di interi

Output: la somma del sottrarray di somma massima

Esempio

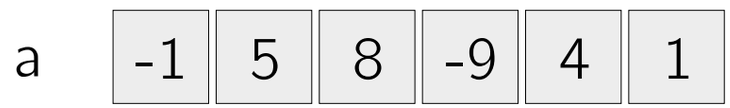
Sottoarray di Somma Massima

Problema: Dato un array a di n interi (positivi e negativi) individuare il sottoarray di somma massima.

Input: un array a di interi

Output: la somma del sottrarray di somma massima

Esempio



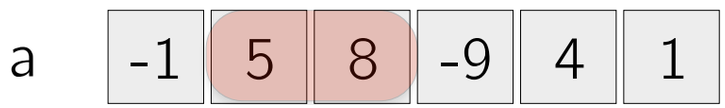
Sottoarray di Somma Massima

Problema: Dato un array a di n interi (positivi e negativi) individuare il sottoarray di somma massima.

Input: un array a di interi

Output: la somma del sottrarray di somma massima

Esempio



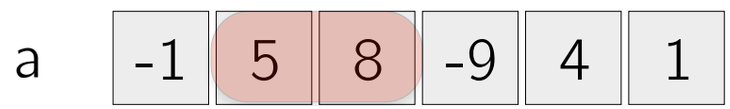
Sottoarray di Somma Massima

Problema: Dato un array a di n interi (positivi e negativi) individuare il sottoarray di somma massima.

Input: un array a di interi

Output: la somma del sottrarray di somma massima

Esempio



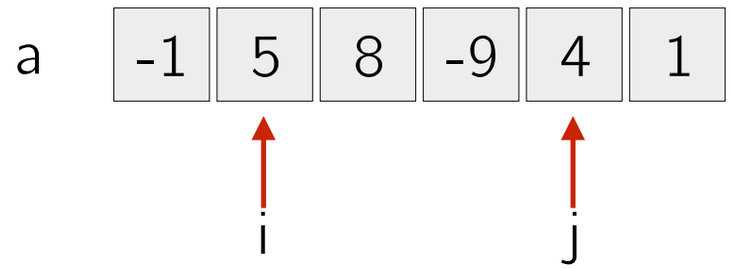
output: 13

Soluzione 1

```
max = a[0];

for(i=0; i<n; i++)
{
    for(j=i; j<n; j++)
    {
        somma=0;
        for(k=i; k<=j; k++)
        {
            somma+=a[k];
        }
        if(somma > max) max=somma;
    }
}
```

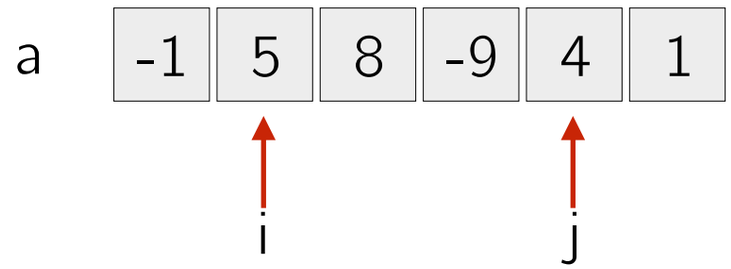
Soluzione 1

```
max = a[0];

for(i=0; i<n; i++)
{
    for(j=i; j<n; j++)
    {
        somma=0;
        for(k=i; k<=j; k++)
        {
            somma+=a[k];
        }
        if(somma > max) max=somma;
    }
}
```



Soluzione 1

```
max = a[0];

for(i=0; i<n; i++)
{
    for(j=i; j<n; j++)
    {
        somma=0;
        for(k=i; k<=j; k++)
        {
            somma+=a[k];        O(1)
        }
        if(somma > max) max=somma;
    }
}
```

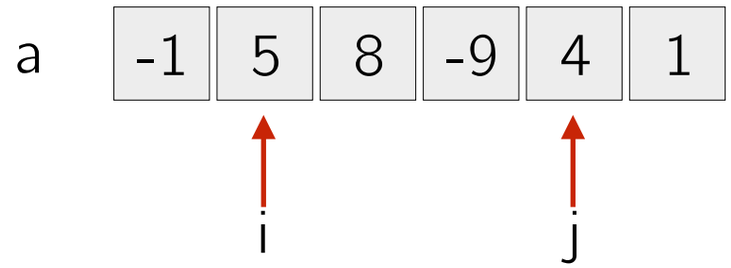


Soluzione 1

```
max = a[0];

for(i=0; i<n; i++)
{
    for(j=i; j<n; j++)
    {
        somma=0;
        for(k=i; k<=j; k++)
        {
            somma+=a[k];
        }
        if(somma > max) max=somma;
    }
}
```

O(1)

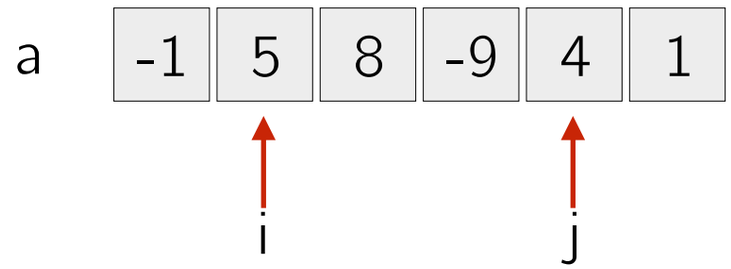
O(n)



Soluzione 1

```
max = a[0];

for(i=0; i<n; i++)
{
    for(j=i; j<n; j++)
    {
        somma=0;
        for(k=i; k<=j; k++)
        {
            somma+=a[k];
        }
        if(somma > max) max=somma;
    }
}
```

O(1) O(n) $O(n^2)$



Soluzione 1

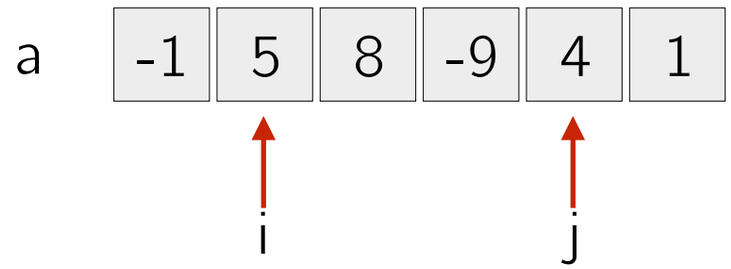
```
max = a[0];

for(i=0; i<n; i++)
{
    for(j=i; j<n; j++)
    {
        somma=0;
        for(k=i; k<=j; k++)
        {
            somma+=a[k];
        }
        if(somma > max) max=somma;
    }
}
```

O(1)

O(n)

O(n²)

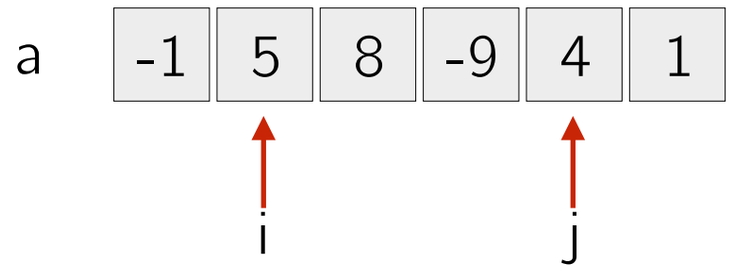
O(n³)



Soluzione 1

```
max = a[0];

for(i=0; i<n; i++)
{
    for(j=i; j<n; j++)
    {
        somma=0;
        for(k=i; k<=j; k++)
        {
            somma+=a[k];
        }
        if(somma > max) max=somma;
    }
}
```

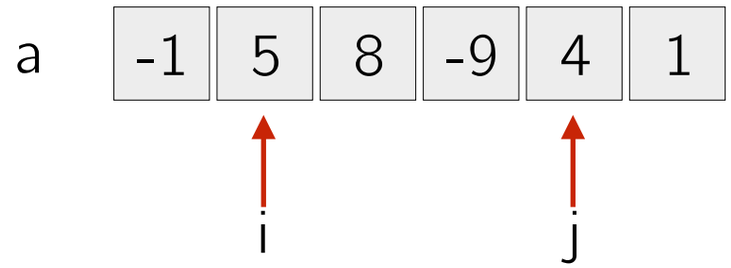
O(1) O(n) $O(n^2)$ $O(n^3)$



Tempo: $O(n^3)$:-(

Soluzione 2

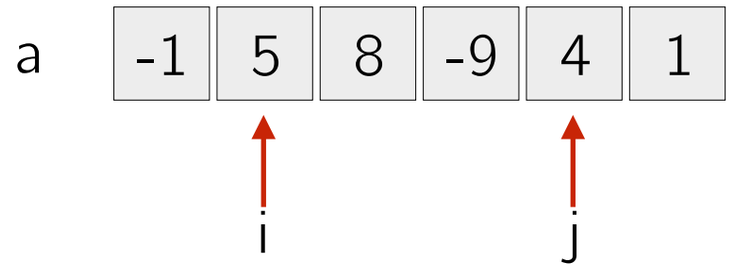
```
max = a[0];

for(i=0; i<n; i++)
{
    somma=0;
    for(j=i; j<n; j++)
    {
        somma+=a[j];
        if(somma > max)
                max=somma;
    }
}
```



Soluzione 2

```
max = a[0];

for(i=0; i<n; i++)
{
    somma=0;
    for(j=i; j<n; j++)
    {
        somma+=a[j];
        if(somma > max)
                max=somma;
    }
}
```

O(1)

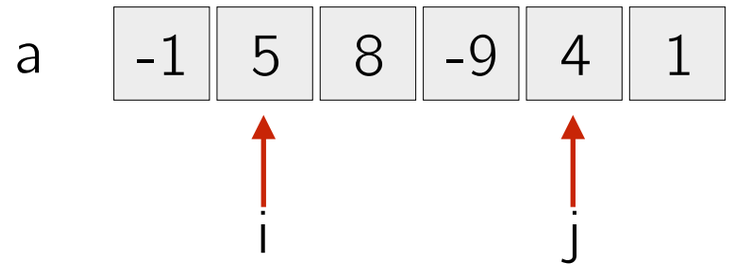


Soluzione 2

```
max = a[0];

for(i=0; i<n; i++)
{
    somma=0;
    for(j=i; j<n; j++)
    {
        somma+=a[j];
        if(somma > max)
                max=somma;
    }
}
```

O(1)

O(n)



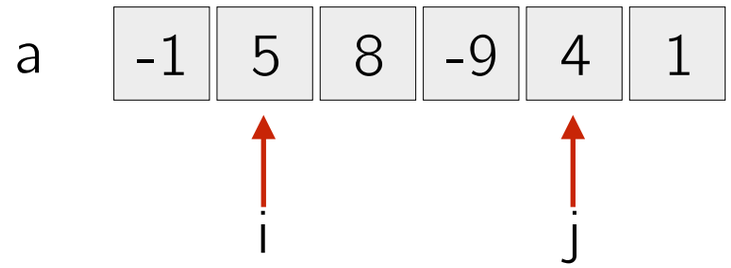
Soluzione 2

```
max = a[0];

for(i=0; i<n; i++)
{
    somma=0;
    for(j=i; j<n; j++)
    {
        somma+=a[j];
        if(somma > max)
                max=somma;
    }
}
```

O(1)

O(n)

$O(n^2)$



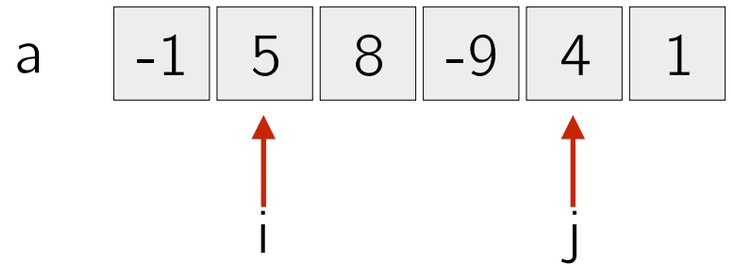
Soluzione 2

```
max = a[0];

for(i=0; i<n; i++)
{
    somma=0;
    for(j=i; j<n; j++)
    {
        somma+=a[j];
        if(somma > max)
            max=somma;
    }
}
```

O(1)

O(n)

O(n^2)



Tempo: O(n^2) :-|

Come fare meglio?

Possiamo sfruttare due proprietà del sottrarray di somma massima

Come fare meglio?

Possiamo sfruttare due proprietà del sottrarray di somma massima

1) La somma dei valori in ogni prefisso del sottoarray ottimo è positiva, se così non fosse potremmo eliminare tale prefisso ottenendo un sottoarray di somma maggiore (assurdo).

Come fare meglio?

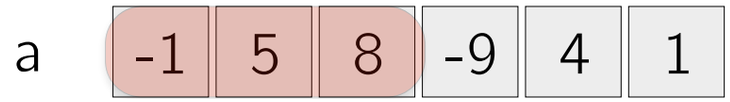
Possiamo sfruttare due proprietà del sottrarray di somma massima

1) La somma dei valori in ogni prefisso del sottoarray ottimo è positiva, se così non fosse potremmo eliminare tale prefisso ottenendo un sottoarray di somma maggiore (assurdo).

Come fare meglio?

Possiamo sfruttare due proprietà del sottrarray di somma massima

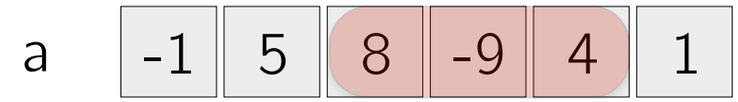
1) La somma dei valori in ogni prefisso del sottoarray ottimo è positiva, se così non fosse potremmo eliminare tale prefisso ottenendo un sottoarray di somma maggiore (assurdo).



Come fare meglio?

Possiamo sfruttare due proprietà del sottrarray di somma massima

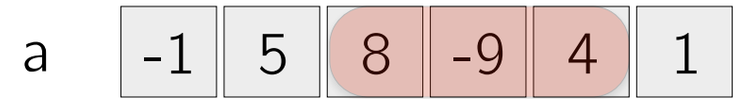
1) La somma dei valori in ogni prefisso del sottoarray ottimo è positiva, se così non fosse potremmo eliminare tale prefisso ottenendo un sottoarray di somma maggiore (assurdo).



Come fare meglio?

Possiamo sfruttare due proprietà del sottrarray di somma massima

1) La somma dei valori in ogni prefisso del sottoarray ottimo è positiva, se così non fosse potremmo eliminare tale prefisso ottenendo un sottoarray di somma maggiore (assurdo).

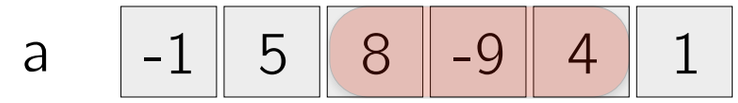


2) Il valore immediatamente precedente al primo valore del sottoarray ottimo è negativo, se così non fosse potremmo aggiungere tale valore ottenendo un sottoarray di somma maggiore (assurdo)

Come fare meglio?

Possiamo sfruttare due proprietà del sottrarray di somma massima

1) La somma dei valori in ogni prefisso del sottoarray ottimo è positiva, se così non fosse potremmo eliminare tale prefisso ottenendo un sottoarray di somma maggiore (assurdo).

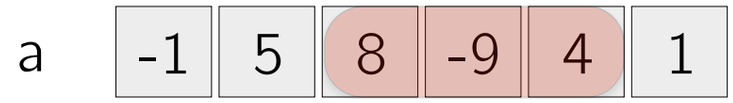


2) Il valore immediatamente precedente al primo valore del sottoarray ottimo è negativo, se così non fosse potremmo aggiungere tale valore ottenendo un sottoarray di somma maggiore (assurdo)

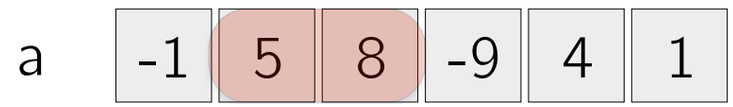
Come fare meglio?

Possiamo sfruttare due proprietà del sottrarray di somma massima

1) La somma dei valori in ogni prefisso del sottoarray ottimo è positiva, se così non fosse potremmo eliminare tale prefisso ottenendo un sottoarray di somma maggiore (assurdo).



2) Il valore immediatamente precedente al primo valore del sottoarray ottimo è negativo, se così non fosse potremmo aggiungere tale valore ottenendo un sottoarray di somma maggiore (assurdo)



Soluzione 3

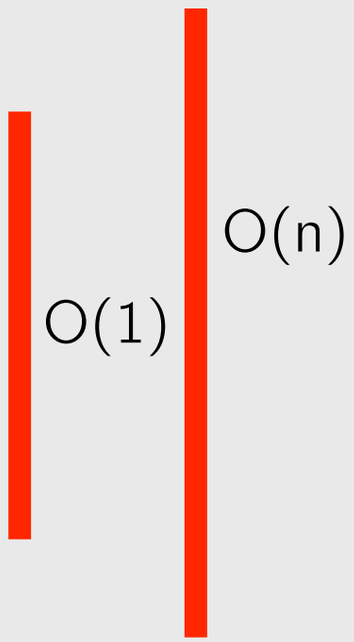
```
max = A[0]; somma = 0;

for(i=0; i<n; i++)
{
   if(somma > 0) somma+=a[i];
   else somma=a[i];

   if(somma > max) max=somma;
}
```

Soluzione 3

```
max = A[0]; somma = 0;

for(i=0; i<n; i++)
{
   if(somma > 0) somma+=a[i];
   else somma=a[i];

   if(somma > max) max=somma;
}
```

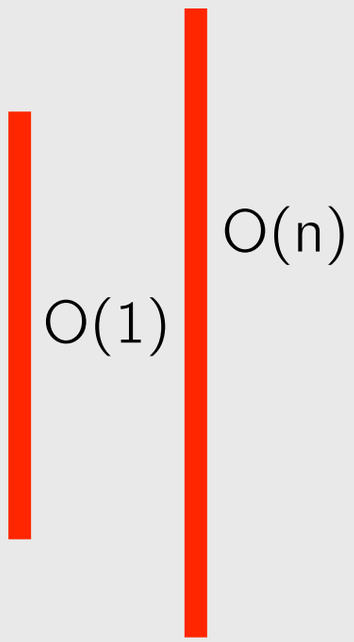
Soluzione 3

```
max = A[0]; somma = 0;

for(i=0; i<n; i++)
{
   if(somma > 0) somma+=a[i];
   else somma=a[i];

   if(somma > max) max=somma;
}
```



Soluzione 3

```
max = A[0]; somma = 0;

for(i=0; i<n; i++)
{
   if(somma > 0) somma+=a[i];
   else somma=a[i];

   if(somma > max) max=somma;
}
```



Tempo: O(n) :-)

Soluzione 3

```
max = A[0]; somma = 0;

for(i=0; i<n; i++)
{
   if(somma > 0) somma+=a[i];
   else somma=a[i];

   if(somma > max) max=somma;
}
```

O(1)

O(n)

somma

-1

-1	5	8	-9	4	1

Tempo: O(n) :-)

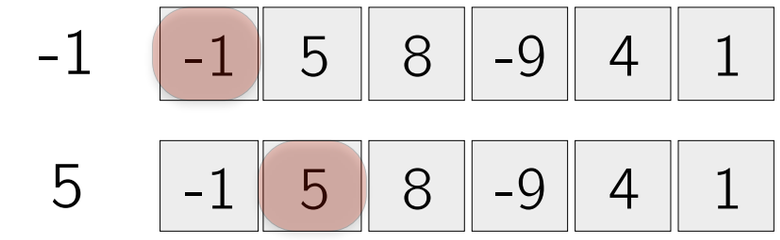
Soluzione 3

```
max = A[0]; somma = 0;

for(i=0; i<n; i++)
{
   if(somma > 0) somma+=a[i];
   else somma=a[i];

   if(somma > max) max=somma;
}
```

O(1)

O(n)

somma



Tempo: O(n) :-)

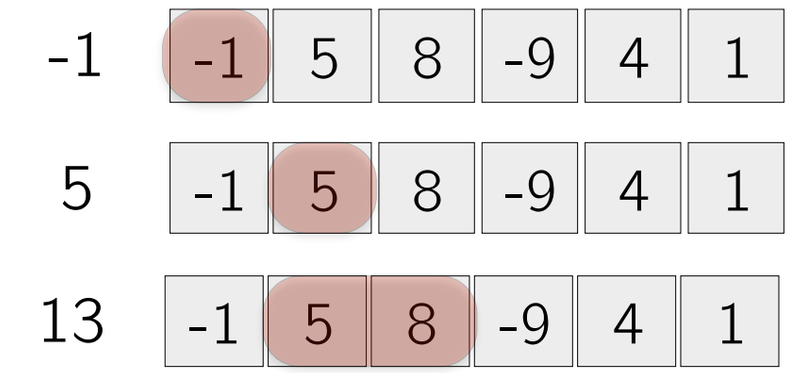
Soluzione 3

```
max = A[0]; somma = 0;

for(i=0; i<n; i++)
{
   if(somma > 0) somma+=a[i];
   else somma=a[i];

   if(somma > max) max=somma;
}
```

O(1)

O(n)

somma



Tempo: O(n) :-)

Soluzione 3

```
max = A[0]; somma = 0;

for(i=0; i<n; i++)
{
   if(somma > 0) somma+=a[i];
   else somma=a[i];

   if(somma > max) max=somma;
}
```

O(1)

O(n)

somma

-1	-1	5	8	-9	4	1
5	-1	5	8	-9	4	1
13	-1	5	8	-9	4	1
4	-1	5	8	-9	4	1

Tempo: O(n) :-)

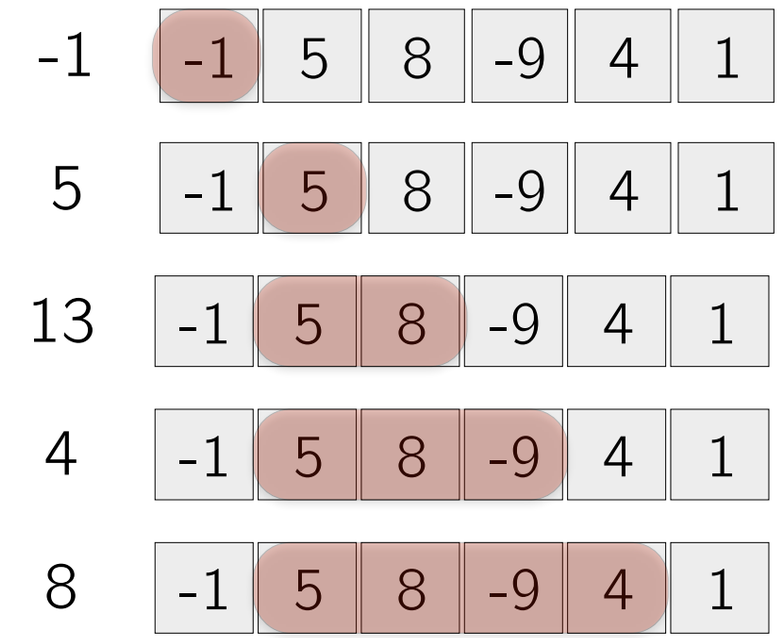
Soluzione 3

```
max = A[0]; somma = 0;

for(i=0; i<n; i++)
{
   if(somma > 0) somma+=a[i];
   else somma=a[i];

   if(somma > max) max=somma;
}
```

O(1)

O(n)

somma



Tempo: O(n) :-)

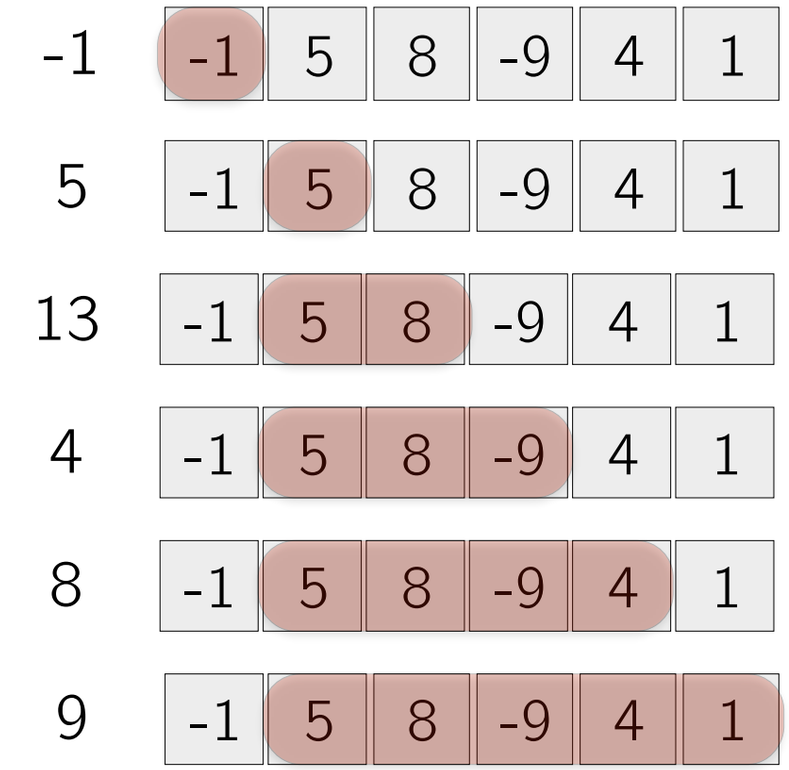
Soluzione 3

```
max = A[0]; somma = 0;

for(i=0; i<n; i++)
{
   if(somma > 0) somma+=a[i];
   else somma=a[i];

   if(somma > max) max=somma;
}
```

O(1)

O(n)

somma



Tempo: O(n) :-)

Esercizio 1

Intersezione tra array

Scrivere un programma che accetti in input due array di interi distinti e restituisca in output il numero di elementi che occorrono in entrambi gli array. Si assuma che la lunghezza di ogni array sia fornita prima dell'immissione degli elementi.

Una volta scritto il codice e superata la verifica sul sito, scaricare e decomprimere il file a questo indirizzo:

`http://didawiki.cli.di.unipi.it/lib/exe/fetch.php/informatica/all-b/test_set.zip`

La directory contiene input diversi (file `.in`) insieme agli output attesi (file `.out`). Ad esempio: `100.in` contiene 2 array di lunghezza cento, ed è possibile usarlo come input al programma utilizzando la redirezione dell'input vista a lezione:

```
./esercizio.o < 100.in
```

Si provi a lanciare il programma su input diversi e per ogni input si controlli che l'output sia giusto confrontandolo col valore nel rispettivo file `.out`, che è possibile stampare sul terminale col comando `cat`, ad esempio:

```
cat 100.out
45
```

Infine, si provi a misurare quanto tempo impiega il programma su input diversi utilizzando il comando `time`, che restituisce in output il tempo impiegato dal programma, ad esempio:

```
time ./esercizio.o < 100.in
45
real 0.066 user 0.005 sys 0.003 pcpu 11.07
```

Come varia il tempo impiegato a seconda della dimensione dell'array?

Esercizio 2

Intersezione tra array ordinati

Scrivere un programma che accetti in input due array di interi distinti e restituisca in output il numero di elementi che occorrono in entrambi gli array. Si assuma che la lunghezza di ogni array sia fornita prima dell'immissione degli elementi.

Si assuma che questa volta gli array vengano inseriti *ordinati in maniera strettamente crescente*. Si può calcolare l'intersezione in maniera più efficiente?

Dopo aver superato i test sul sito, si ripetano gli esperimenti sul dataset utilizzato nel precedente esercizio (gli array vengono forniti in ordine crescente) e si confrontino i risultati ottenuti.

`http://didawiki.cli.di.unipi.it/lib/exe/fetch.php/informatica/all-b/test_set.zip`

Esercizio 3

Sottoarray di somma massima

Dato un array A di n interi (positivi e negativi), scrivere un programma che identifichi il sottoarray di A i cui elementi hanno somma massima tra tutti gli altri sottoarray di A e ne stampi la somma.

Una volta scritto il codice e superata la verifica sul sito, scaricare i files di test diponibili all'indirizzo

`http://didawiki.cli.di.unipi.it/lib/exe/fetch.php/informatica/all-b/test_maxsum.zip`

e valutare l'efficienza della propria soluzione.

La soluzione implementata dovrebbe avere complessità lineare, in base all'algoritmo visto a lezione. Come utile esercizio facoltativo, implementare una soluzione di complessità non ottimale (ad esempio quadratica), misurare il tempo impiegato e notare la differenza tra le diverse soluzioni.

Esercizio 4

Fusione di array ordinati

Scrivere un programma che accetti in input due array *ordinati* di interi e restituisca in output l'unione ordinata dei due array. Si assuma che la lunghezza di ogni array sia fornita prima dell'immissione degli elementi.

Per semplicità si assuma che l'intersezione tra i due array sia vuota.

Puzzle

L'intero mancante

Viene dato un array di dimensione N contenente (non ordinati) tutti gli interi compresi tra 1 e $N + 1$ ad eccezione di uno di essi e si vuole stabilire l'elemento mancante.

Sono possibili almeno 4 soluzioni aventi le seguenti complessità:

Tempo	Spazio aggiuntivo	
$O(n^2)$	$O(1)$	`:-((`
$O(n \log n)$	$O(n)$	`:-(`
$O(n)$	$O(n)$	`:-)`
$O(n)$	$O(1)$	`:-D`

Input

```
8
3
4
9
2
7
1
8
6
```

Output

```
5
```