Si implementi il tipo di dato astratto `BitArray`, che memorizza l'informazione contenuta in un vettore di booleani all'interno di singoli bit di una variabile intera senza segno della stessa lunghezza, al fine di minimizzare l'uso della memoria. Nell'implementare le funzionalità elencate sotto, tenere presente che le operazioni sul tipo `BitArray` debbono essere il più veloci possibile. Pertanto viene espressamente richiesto di utilizzare le operazioni bit-a-bit tra tipi unsigned ovunque possibile, poiché è noto essere estremamente veloci grazie al supporto hardware fornito dall'unità aritmetico-logica (ALU).

Per quanto concerne la presente sessione d'esame, si supponga inoltre che un vettore di bit non possa avere una lunghezza maggiore di 32.

E' VIETATO L'USO DI VETTORI IN CUI LA SINGOLA COMPONENTE RAPPRESENTI IL SINGOLO BIT, QUALI, AD ESEMPIO, I VETTORI DI BOOL.

Si implementi il tipo di dato astratto `BitArray`, fornendo le seguenti funzionalità:

--- Metodi invocati nella PRIMA PARTE di main.cpp: ---

- ✓ **`BitArray a(vetBool, len);`**
  Costruttore che crea un `BitArray` a di lunghezza `len`, inizializzato secondo il contenuto del vettore di booleani `vetBool`.
- ✓ **`cout<<s;`**
  Operatore di uscita per il tipo `BitArray`. L'uscita ha la seguente forma:

  ```
  [T,F,F,T,T,T]
  ```

  dove viene riportata la sequenza di valori booleani secondo la usuale rappresentazione estesa. Gli estremi dell'array sono delimitati da parentesi quadre ***senza*** alcuno spazio tra le stesse e il più vicino elemento dell'array. L'operatore ***non*** aggiunge il *new line* alla fine.
- ✓ **`!a;`**
  Operatore che restituisce il numero di elementi a **1** all'interno del `BitArray` `a`.
- ✓ **`a1|a2;`**
  Operatore che restituisce un ***nuovo*** `BitArray` inizializzato al risultato dell'OR bit a bit tra `a1` e `a2`. Qualora le lunghezze dei due `BitArray` fossero diverse, il `BitArray` risultante avrebbe lunghezza pari alla massima delle due e (***solo***) per l'operazione di OR il più corto dei due `BitArray` deve essere considerato esteso mediante l'aggiunta in coda di valori **0.**

--- Metodi invocati nella SECONDA PARTE di main.cpp: ---

- ✓ **`a.setBit(start, end, val);`**
  Operazione che setta i bit di `a` dall'indice `start` all'indice `end` (inclusi) al valore `val`. La funzione restituisce `true` se gli input sono corretti, `false` altrimenti.
- ✓ **`a.flip(start, end);`**
  Operazione che nega il valore di tutti i bit dall'indice `start` all'indice `end` (inclusi). . La funzione restituisce `true` se gli input sono corretti, `false` altrimenti.
- ✓ **`a.maxSubSeq();`**

  Operazione che restituisce la lunghezza della più lunga sottosequenza di **0** consecutivi all'interno del `BitArray`.

Mediante il linguaggio C++, realizzare il tipo di dato astratto definito dalle precedenti specifiche. Non è permesso utilizzare funzionalità della libreria STL come il tipo `string`, il tipo `vector`, il tipo `list`, ecc.

```
// file main.cpp

#include "compito.h"
int  main(){

    cout<<"--- PRIMA PARTE ---" <<endl;
    cout << "Test del costruttore (deve stampare [T,F,T,T])" <<endl;
    bool input1[] = {true, false, true, true};
    BitArray a1(input1, 4);
    cout<<a1<<endl<<endl;

    cout << "Altro test del costruttore (deve stampare [F,T,T])" <<endl;
    bool input2[] = {false, true, true};
    BitArray a2(input2, 3);
    cout<<a2<<endl<<endl;

    cout << "Test dell'operatore di negazione logica (deve stampare 3)" <<endl;
    cout<<!a1<<endl<<endl;

    cout << "Test dell'operatore di or bit a bit (deve stampare [T,T,T,T]) " <<endl;
    cout<<(a1|a2)<<endl<<endl;


    cout<<"--- SECONDA PARTE ---" <<endl;
    cout << "Test della funzione flip (deve stampare [T,T,F,T])" <<endl;
    a1.flip(1,2);
    cout<<a1<<endl<<endl;

    cout << "Doppio test della setBit (deve stampare [F,T,T] e [T,T,T])" <<endl;
    a2.setBit(1,3, false);
    cout<<a2<<endl;
    a2.setBit(0,2, true);
    cout<<a2<<endl<<endl;

    cout << "Doppio test della maxSubSeq (deve stampare 1 e poi 4)" <<endl;
    cout<<a1.maxSubSeq()<<endl;
    bool input3[] = {true,false,false,true,false,false,false,false,true};
    BitArray a3(input3, 9);
    cout<<a3.maxSubSeq()<<endl<<endl;

}
```

USCITA ATTESA

```
--- PRIMA PARTE ---
Test del costruttore (deve stampare [T,F,T,T])
[T,F,T,T]

Altro test del costruttore (deve stampare [F,T,T])
[F,T,T]

Test dell'operatore di negazione logica (deve stampare 3)
3

Test dell'operatore di or bit a bit (deve stampare [T,T,T,T])
[T,T,T,T]

--- SECONDA PARTE ---
Test della funzione flip (deve stampare [T,T,F,T])
[T,T,F,T]

Doppio test della setBit (deve stampare [F,T,T] e [T,T,T])
[F,T,T]
[T,T,T]

Doppio test della maxSubSeq (deve stampare 1 e poi 4)
1
4
```