ESERCIZIO 4 DEL II APPELLO 15/16 (4/2/16)

int check (int a[], int dima, int b[], int dimb)

verifica la formula

a[i] è diverso da TUTTI gli elementi di b

$$\exists i \in [0, dima).\Big(\big(\forall j \in [0, dimb).\ a[i] \neq b[j]\big) \wedge \#\{k \mid k \in [0, dima) \wedge a[i] = a[k]\} = 1\Big)$$

il numero degli elementi di a uguali ad a[i] deve essere 1

(dato che k può essere uguale a i questo significa che a[i] è diverso da TUTTI gli altri elementi di a)

k k

[3, 4, 7, 2, 4]

i

$$\#\{k \mid a[i] == a[k]\} = 2$$

Scriviamo intanto una funzione per

$$\forall j \in [0, dimb).\ a[i] \neq b[j]$$

```
int diversi (int b[], int dimb, int x) {
    int j=0;
    int ok=1;
    while (j<dimb && ok) {
        if (b[j]==x) ok=0;
```

```
        j++;
    }
    return ok;
}
```

Scriviamo una funzione per

$$\#\{k \mid k \in [0, dima) \wedge a[i] == a[k]\} = 1$$

```
int unico(int a[], int dima, int i) {
    int cont = 0;
    for (int k=0; k<dima; k++)
        if (a[k]== a[i]) cont++;
    return cont==1;
}
```

⇖ TRADUZIONE "MECCANICA" DELLA FORMULA

Visto la formula mi chiede in sostanza di vedere se a[i] è unico posso in realtà scrive la funzione come segue:

```
int unico(int a[], int dima, int i) {
    int j=0;
    int ok=1;
    while (j<dima && ok) {
        if (i!=j && a[i]==a[j])
            ok=0;
        j++;
    }
    return ok;
}
```

⇖

$$\forall j \in [0, dima).\ (i == j \vee a[i] != a[j])$$

mi rimane da verificare

$$\exists i \in [0, dima) \,.\, divisi(b, dimb, a[i]) \wedge unico(a, dima, i)$$

```
int check (int a[], int dima, int b[], dimb) {
    int i=0;
    int Trovato = 0;
    while (i < dima && !Trovato) {
        if ( divisi (b, dimb, a[i]) && unico(a, dima, i))
            Trovato = 1;
    } i++;
    return Trovato;
}
```

ESERCIZIO 2 I COMPITINO 2016/2017 (31/10/17)

int contaUnico (int a[], int b[], int dima, int dimb)

restituisce il numero di elementi diversi di a che sono contenuti in b

a = [60; 20; 30; 10; 20; 40; 60; 80]

b = [60; 30; 30; 10; 20; 60]

risultato = 4 {60, 20, 30, 10}

usiamo la member già vista a lezione

```
int member (int b[], int dimb, int x) {
    int i=0;
    int Trovato=0;
    while (i<dimb && !Trovato) {
        if (b[i]==x) Trovato=1;
        i++;
    }
    return Trovato;
}
```

idea per evitare di controllare più volte la presenza dello stesso numero: controllo che sia la prima occorrenza in a

a[10; 30; 40; 10; 20;]

SE L'ELEMENTO NON È PRESENTE NELLA PORZIONE DI a CHE LO PRECEDE CHIAMO MEMBER PER VEDERE SE È IN b

CON UNA FUNZIONE

```
int cerca(int a[], int dim, int i) {
    int j = 0;
    int Trovato = 0;
    while (j < i && !Trovato) {
        if (a[j] == a[i]) Trovato = 1;
        j++;
    }
    return Trovato;
}
```

```
int contaUnico(int a[], int b[], int dima, int dimb) {
    int cont=0;
    for (int i=0; i<dima; i++) {
        if (!cerca(a, dima, i))
            if (member(b, dimb, a[i]))
                cont++;
    }
    return cont;
}
```

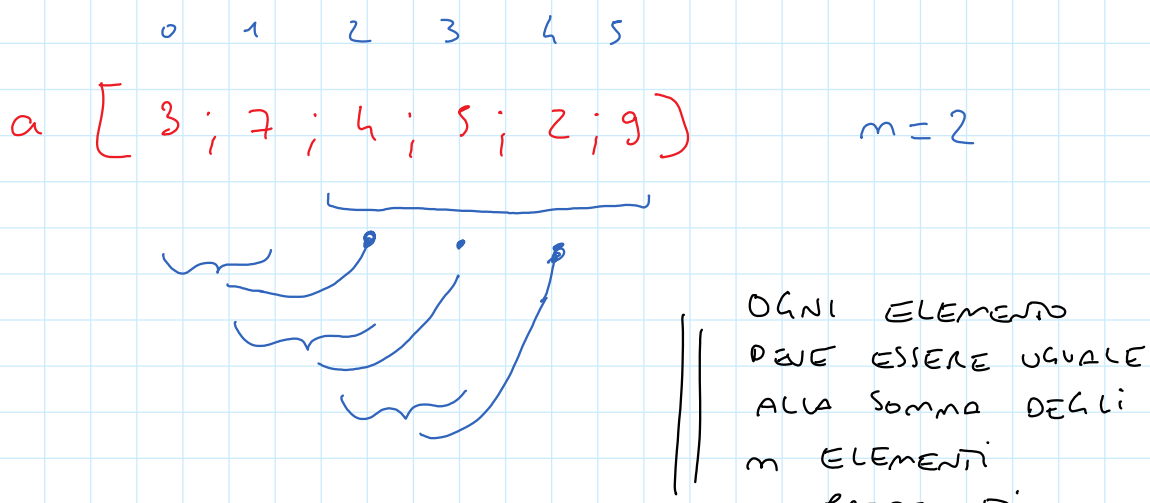
Se è la prima occorrenza

ed è presente in b

ESERCIZIO 1 I APPELLO 2016/2017 (19/01/2017)

dato un array a di dimensione dim e un numero $n \in \mathbb{N}$ t.c. $1 \leq n \leq dim$ verificare

$$\forall j \in [n, dim). \left(a[j] = \sum_{i \in [j-n, j)} a[i] \right)$$



INIZIAMO DA IMPLEMENTARE UNA FUNZIONE PER

$$\sum_{i \in [j-n, j)} a[i]$$

inizio fine

```
int Somma ( int a[], int dim, int inizio,
                                  int fine)
  int s = 0;
  for ( int i = inizio ; i < fine ; i++)
      s += a[i];
```

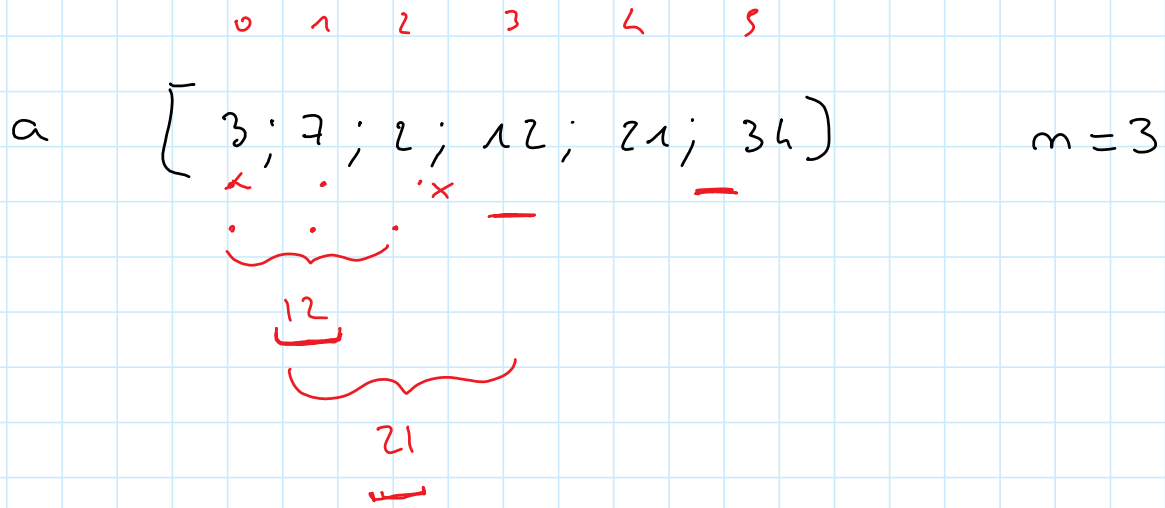
return s;

}

mi rimane quindi da implementare

$$\forall j \in [m, dim) . \left(a[j] == somma(a, dim, j-m, j)\right)$$

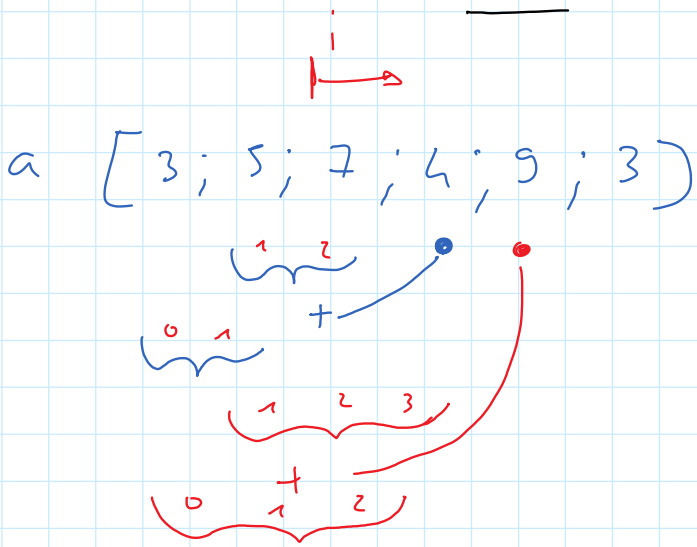
```
int check (int a[], int dim, int m) {
    int j = m;
    int ok = 1;
    while (j < dim && ok) {
        if (a[j] != somma (a, dim, j-m, j))
            ok = 0;
        j++;
    }
    return ok;
}
```



POTREI DEFINIRE UNA SOLUZIONE PIÙ EFFICIENTE CALCOLANDO LA SOMMA A PARTIRE DALLA SOMMA CALCOLATA AL PASSO PRECEDENTE.

ESERCIZIO 1 II APPELLO 2016/2017 (9/2/17)

$$\forall i \in [2, dim).\ a[i] = \left(\sum_{j \in [1, i-1]} a[j]\right) + \left(\sum_{k \in [0, i-2]} a[k]\right)$$



RIUTILIZZO LA FUNZIONE SOMMA VISTA PRIMA CHE SOMMAVA GLI ELEMENTI DI INDICI [inizio, fine)

$[1, i-1] = [1, i)$ $[0, i-2] = [0, i-1)$ ← ESCLUSO

$$\forall i \in [2, dim).\ a[i] == Somma(a, dim, 1, i) + Somma(a, dim, 0, i-1)$$

⇒ solita funzione con ciclo while e variabile OK ...