Lezione 2 II semestre

- esercizio 1
- for
- esercizio 2
- estensioni

esercizio 1

0) leggere N ➔ *cin >> N*

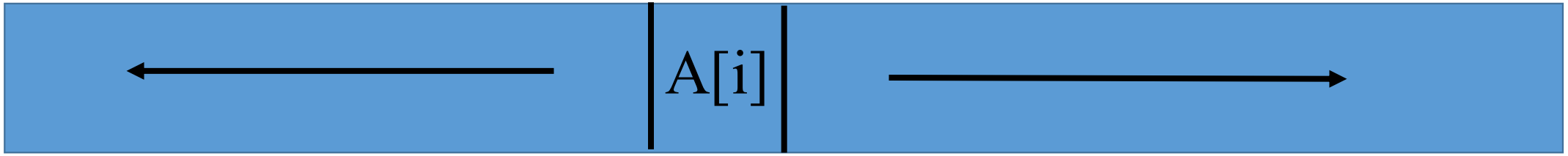
1) leggere N interi in A

```
int i=0;
while(i < N)
        {cin >> A[i]; i=i+1;}
```

2) trovare le duplicazioni in A : *2 modi*

3) mettere le duplicazioni in B in modo che ogni valore duplicato compaia una volta sola

2) trovare le duplicazioni:



sono ripetute le occorrenze dalla seconda in poi

sono ripetute le occorrenze dalla prima alla penultima

entrambe possono andare, ma per i test dobbiamo scegliere una delle 2 per avere tutti lo stesso ordine delle ripetizioni

le consideriamo entrambe e scriviamo l'invariante per ciascuna

int i=0; while (i <N) {...corpo...}

i) R← =(0<=i<=N) && (b sono i valori diversi in A[0..i-1] che hanno un duplicato alla sinistra, e B[0..b-1] contiene questi valori nell'ordine in cui sono trovati)

ii) R→ =(0<=i<=N) && (b sono i valori diversi di A[0..i-1] che hanno almeno un duplicato alla loro destra in A[0..N-1] e B[0..b-1] contiene questi valori nell'ordine in cui sono trovati)

alla fine i=N e quindi in entrambi i casi, b sarà il totale dei valori replicati diversi che sarà lo stesso numero b
quello che cambia è l'ordine in B dei duplicati

← sarà l'ordine delle seconde occorrenze dei valori replicati

→ sarà l'ordine delle prime occorrenze dei valori replicati

esempio: 2 1 1 2 2 1
per ← 1 2 e per → 2 1

R← =(0<=i<=N) && (b sono i valori diversi in A[0..i-1] che hanno un duplicato alla sinistra, e B[0..b-1] contiene questi valori nell'ordine in cui sono trovati)

nel corpo del ciclo dobbiamo
a) decidere se il prossimo elemento di A ripete
b) in caso che sia ripetuto, dobbiamo decidere se è la seconda istanza del valore ripetuto oppure una successiva

a)

```
bool rip=false; int j=0;
while (j <i && rip==false) // R1
{
  if(A[i]==A[j])
    rip=true;
  i=i+1;
}

//R1=(0<=j<=i) && (rip sse A[i] è in A[0..j-1])
```

b)

```
j=0; bool trovato=false;
while(j < b && ! trovato) //R2
{
   if (A[i] ==B[j])
        trovato=true;
}
```

//R2=(0<=j<=b) &&(trovato sse A[i] è in B[0..j-1])

possiamo mettere insieme i diversi pezzi

```
int N, A[100],B[100],b=0, i=1;
while (i<N)//R← =(0<=i<=N) && (b sono i valori diversi in A[0..i-1] che hanno un
duplicato alla sinistra, e B[0..b-1] contiene questi valori nell'ordine in cui sono trovati)
  {
   bool rip=false; int j=0;
   while(j<i && ! rip) //R1
a)        {if(A[j]==A[i])  rip=true;
          j=j+1;}
            if(rip)
            { bool trovato=false; j=0;
              while ( j<b && !trovato)//R2
b)              {if (A[i]==B[j]) trovato = true;
                 j=j+1; }
              if(!trovato)
               { B[b]=A[i];
                 b=b+1;
               }
            }
   i=i+1; }
```

correttezza del ciclo esterno

```
 if(b==0)
    cout<<"Tutto regolare"<<endl;
  else
  {
    int i=0;
    while (i<b)
      {cout<< B[i]<<endl; i=i+1;}
   }
}
```

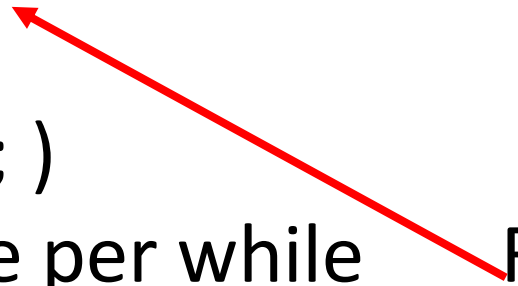
while e for

```
int i=0;
while (i<b)
    {cout<< B[i]<<endl; i=i+1;}
```

```
for(int i=0; i < b ; i=i+1)
   cout << B[i] << endl;
```

```
for( dic. indici ; condiz. di permanenza ; incremento)
```

anche for(; true;)
correttezza come per while



R

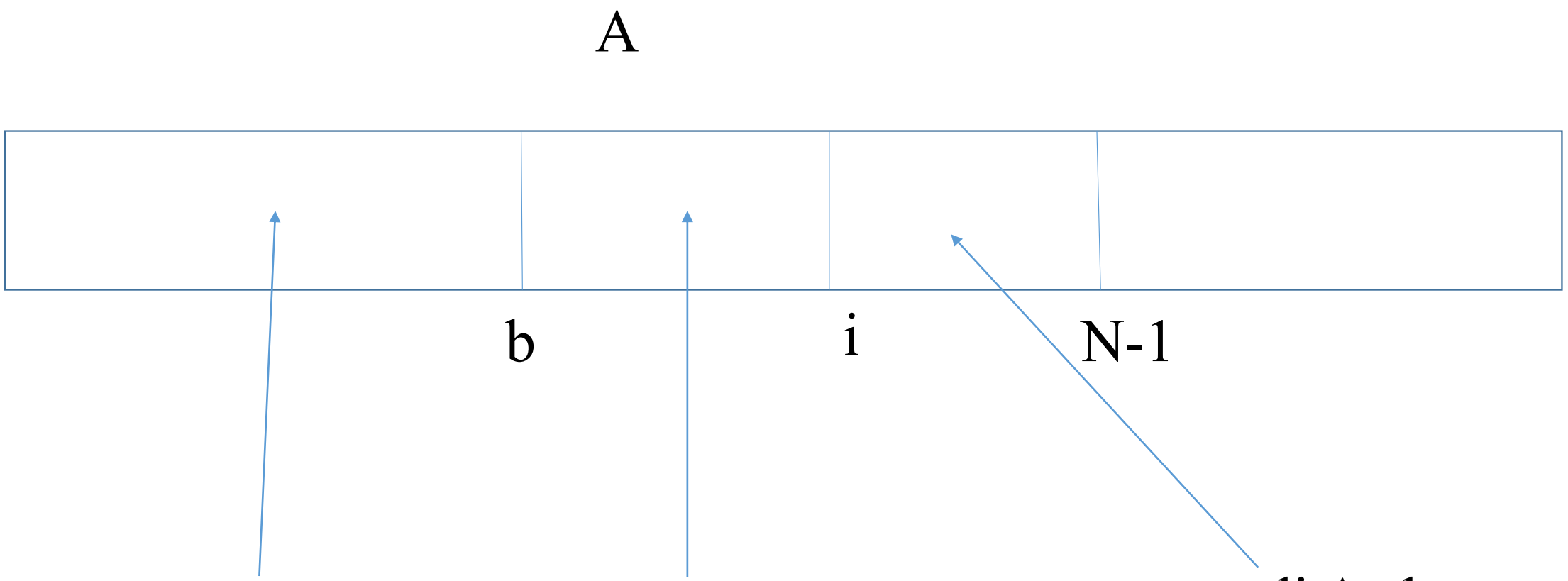
```
 int N, A[100],B[100],b=0;
for(int i=1; i<N; i=i+1)//R←
   {
     bool rip=false;
     for(int j=0; j<i && ! rip;j=j+1)//R1
              if(A[j]==A[i])
                rip=true;
     if(rip)
              {
                bool trovato=false;
                for(int j=0; j<nd && !trovato; j=j+1)//R2
                  if (A[i]==D[j])
                    trovato = true;
                if(!trovato)
                  {
                    B[b]=A[i];
                    b=b+1;
                  }
            }
  }
```

```
 if(b==0)
    cout<<"Tutto regolare"<<endl;
  else
    for(int i=0; i<b; i++)
      cout<< B[i]<<endl;
}
```

esercizio 2

si tratta di eliminare le duplicazioni mantenendo solo la prima copia di ciascun valore

a) scorrere l'array
b) scoprire se l'elemento corrente è ripetuto o no
c) per fare (b) ci serve mantenere i valori diversi incontrati fino a quel momento
d) da (c) alla fine avremo solo gli elementi diversi esistenti nell'array iniziali, come richiesto


A
b
i
N-1
valori diversi incontrati finora
elementi ripetuti che vanno eliminati
parte di A che resta da esaminare

A cambia durante l'esecuzione, quindi chiamiamo vA (v=vecchio) il suo valore iniziale

invariante:
R= (0<=i<=N) && (vA[0..i-1] esaminato e A[0..b-1] sono i valori diversi in vA[0..i-1] nell'ordine richiesto)&&(A[i..N-1]=vA[i..N-1])

inizialmente i=0 e b=0

--per sapere se A[i] è nuovo basta confrontarlo con A[0..b-1]
----se è nuovo lo mettiamo in A[b] e andiamo avanti, altrimenti andiamo avanti

```
for(int i=0; i< N; i++)// R
  {
    bool buono=true;
    for(int j=0; j<b && buono; j++)//R1
       if(A[i]==A[j])
         buono=false;
    if(buono)
       {A[b]=A[i]; b++;}
  }
```

R1=(0<=j<=b)&&(buono sse A[i] è in A[0..j-1])