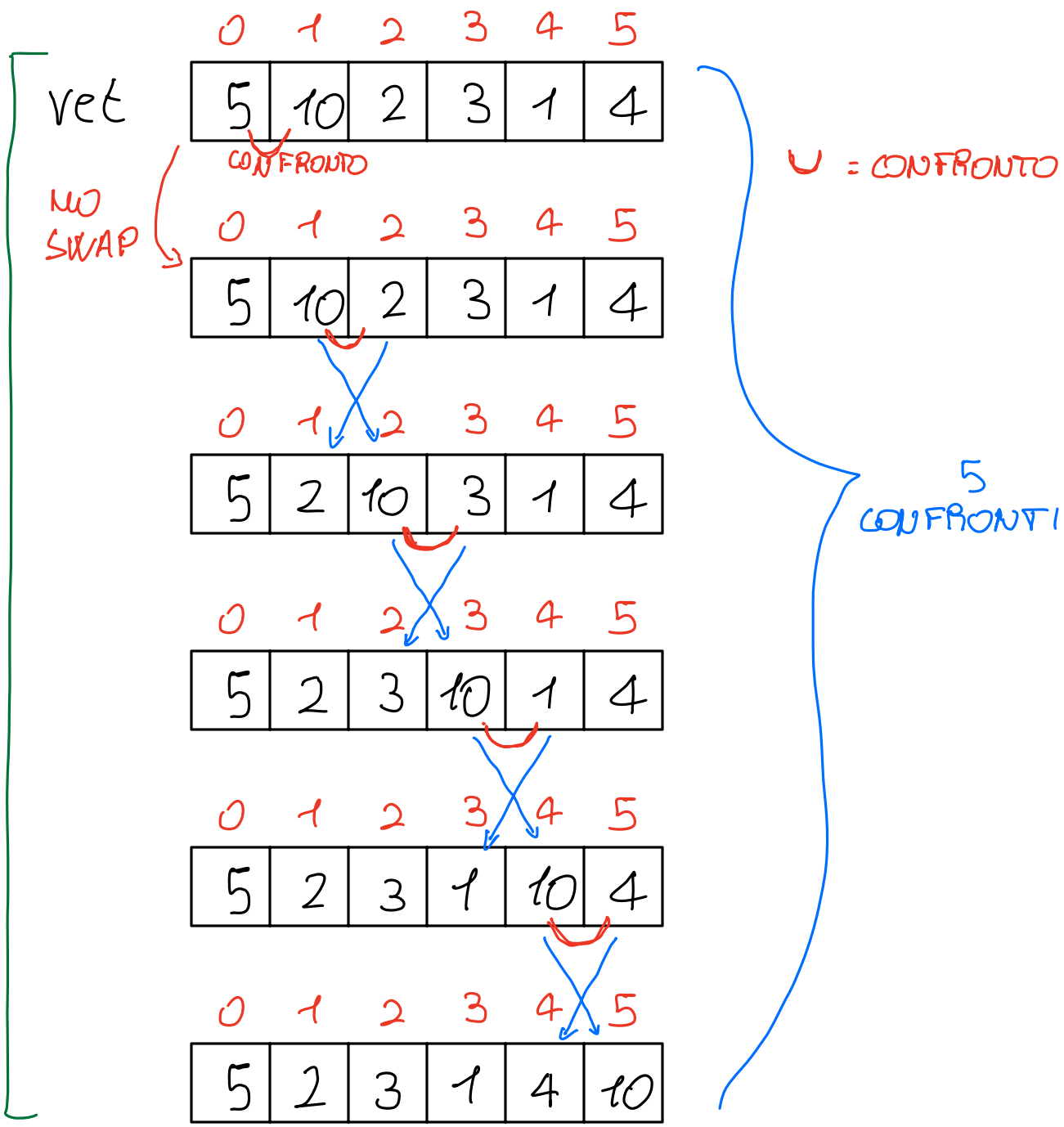
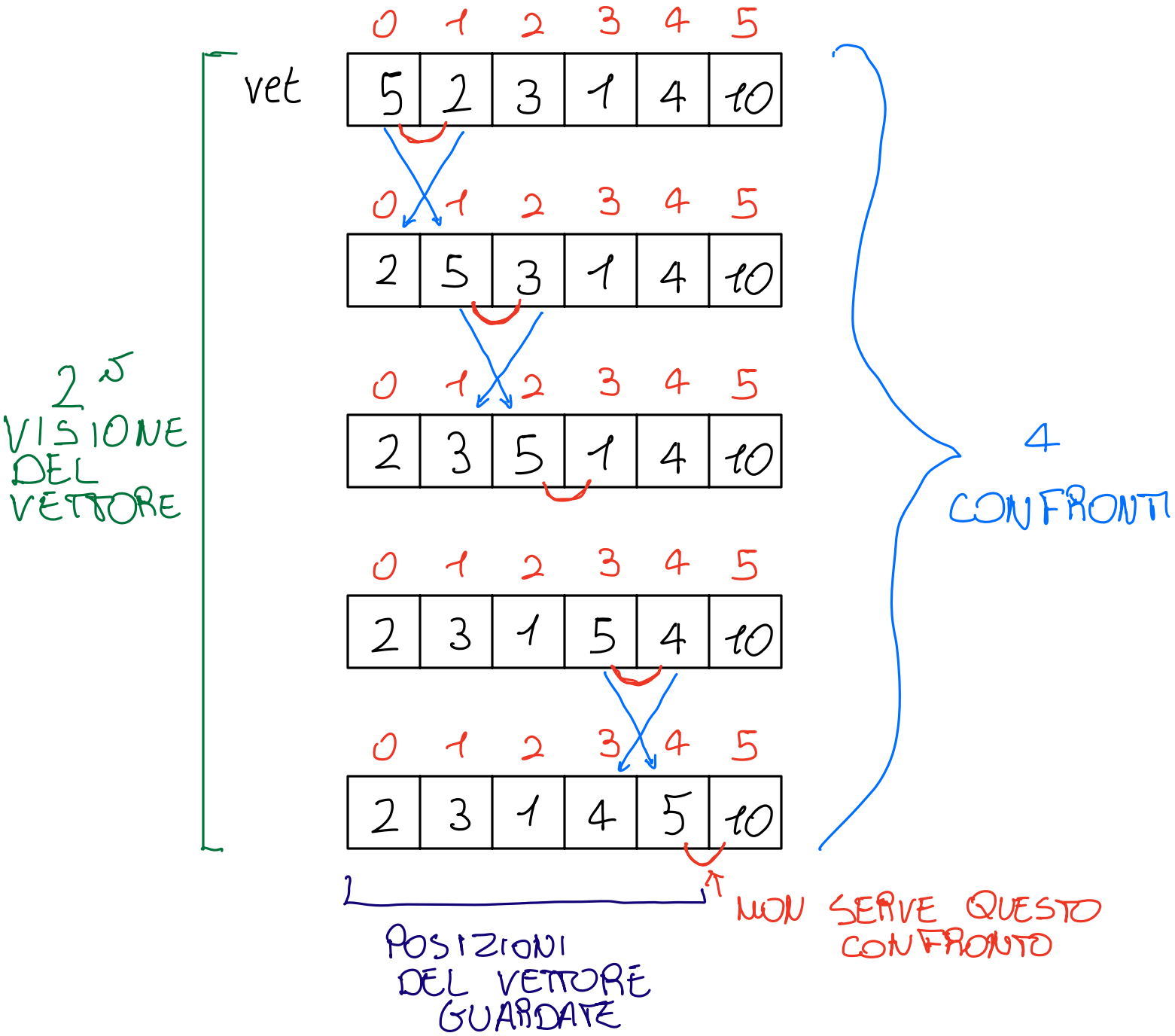
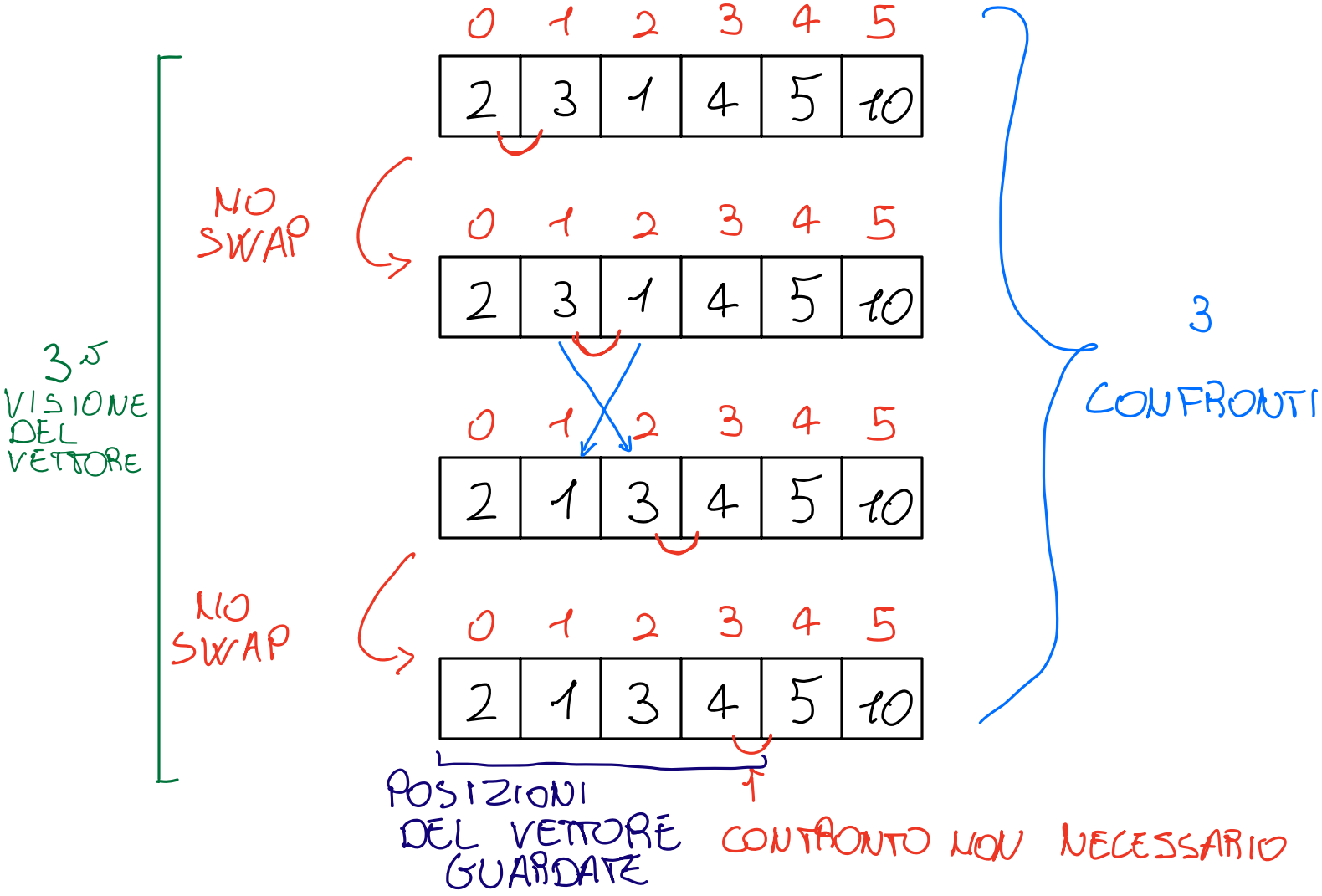
BUBBLE SORT

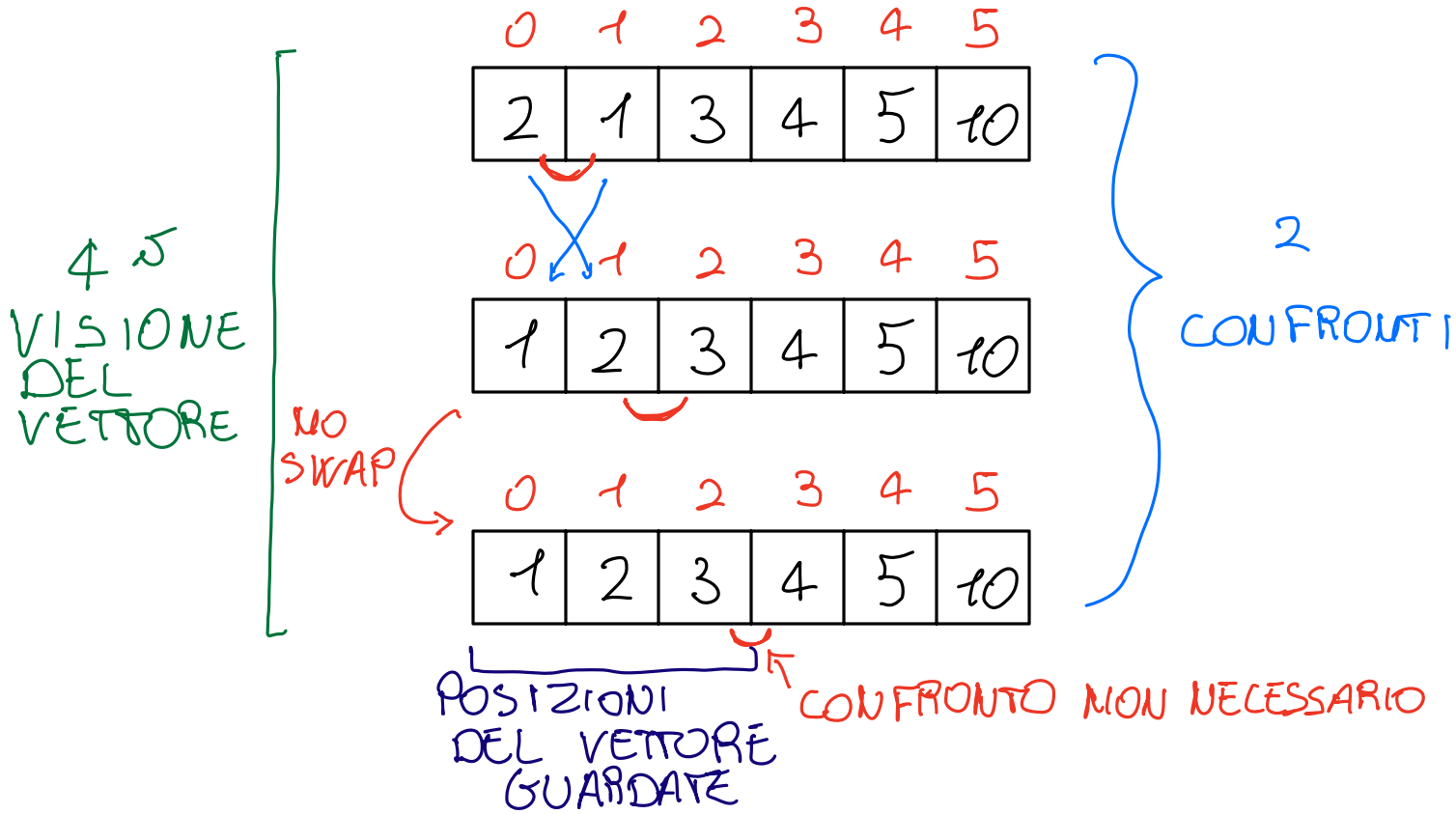
PRIMA DEL main() → #define DIM 6

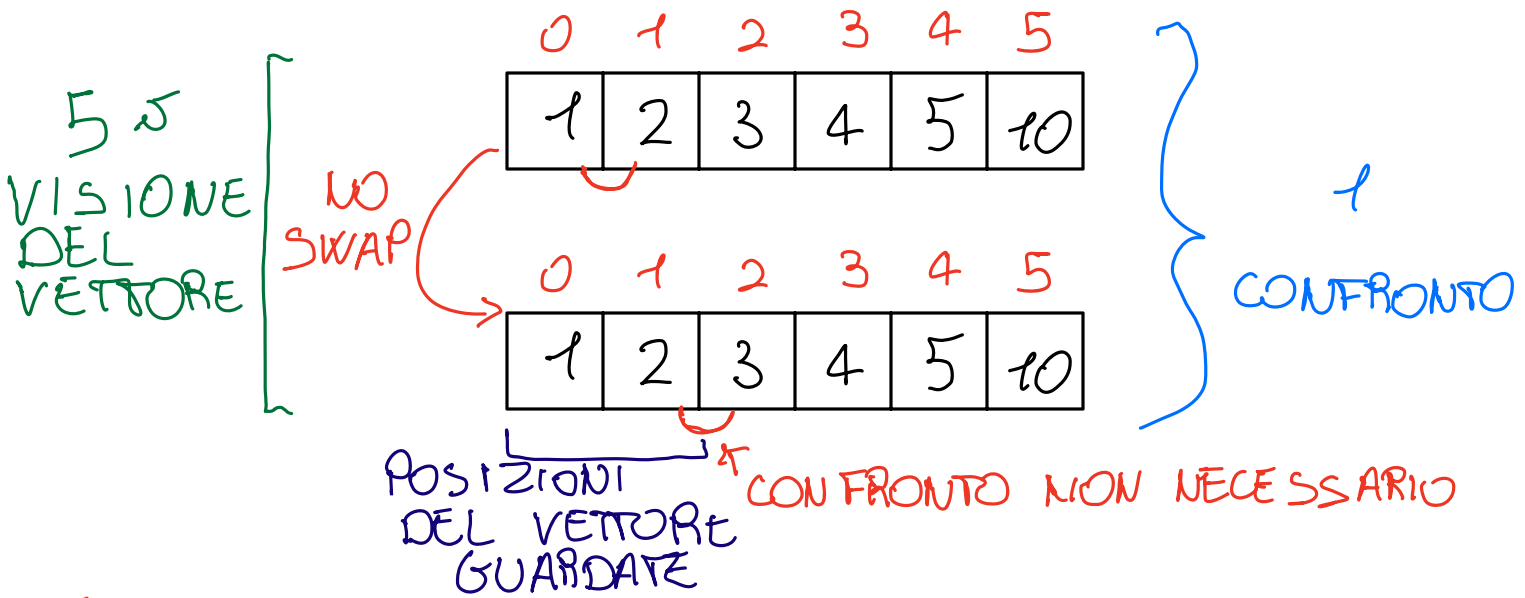
NEL main() → int vet[DIM] = { 5, 10, 2, 3, 1, 4 };

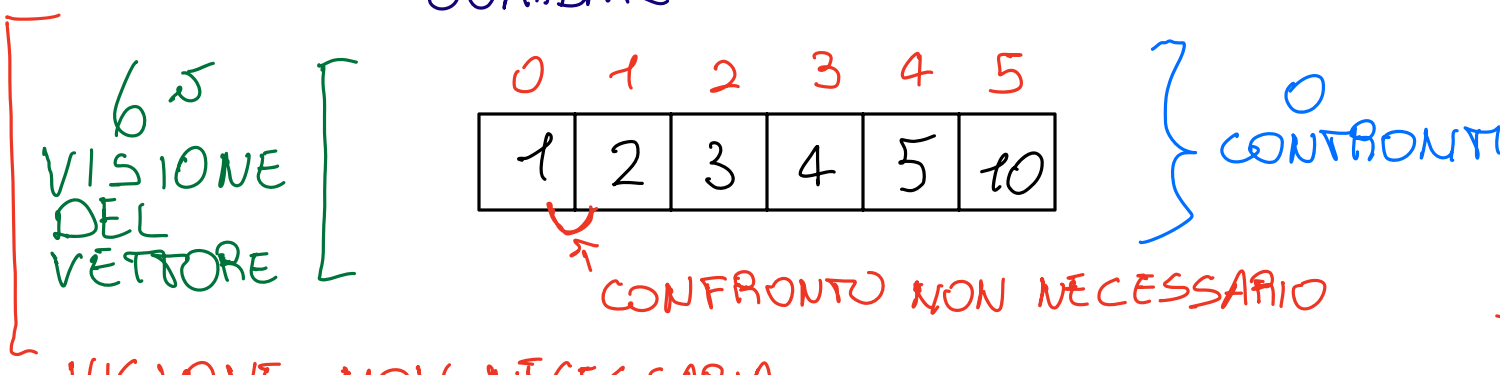



0 1 2 3 4 5
vet
5 2 3 1 4 10
0 1 2 3 4 5
2 5 3 1 4 10
0 1 2 3 4 5
2 3 5 1 4 10
0 1 2 3 4 5
2 3 1 5 4 10
0 1 2 3 4 5
2 3 1 4 5 10
2 5
VISIONE DEL VETTORE
4
CONFRONTI
NON SERVE QUESTO CONFRONTO
POSIZIONI DEL VETTORE GUARDATE


0 1 2 3 4 5
2 3 1 4 5 10
NO SWAP
0 1 2 3 4 5
2 3 1 4 5 10
0 1 2 3 4 5
2 1 3 4 5 10
NO SWAP
0 1 2 3 4 5
2 1 3 4 5 10
3 VISIONE DEL VETTORE
3 CONFRONTI
POSIZIONI DEL VETTORE GUARDATE
CONFRONTO NON NECESSARIO


0 1 2 3 4 5
2 1 3 4 5 10
0 1 2 3 4 5
1 2 3 4 5 10
0 1 2 3 4 5
1 2 3 4 5 10
4 ^
VISIONE DEL VETTORE
NO SWAP
2
CONFRONTI
POSIZIONI DEL VETTORE GUARDATE
CONFRONTO NON NECESSARIO


0 1 2 3 4 5
1 2 3 4 5 10
0 1 2 3 4 5
1 2 3 4 5 10
5 ^
VISIONE DEL VETTORE
NO SWAP
1
CONFRONTO
POSIZIONI DEL VETTORE GUARDATE
CONFRONTO NON NECESSARIO


0 1 2 3 4 5
1 2 3 4 5 10
6 ^
VISIONE DEL VETTORE
0
CONFRONTI
CONFRONTO NON NECESSARIO
VISIONE NON NECESSARIA

FUNZIONE BUBBLE SORT

```
void bubbleSort(int vet[], int dim){
    int i, j;
    int tmp;

    for(i=0; i<(dim-1); i++){
        for(j=0; j<(dim-i-1); j++){
            if(vet[j] > vet[j+1]){
                tmp = vet[j];
                vet[j] = vet[j+1];
                vet[j+1] = tmp;
            }
        }
    }
}
```

PER L'ORDINE CRESCENTE

INSERTION SORT

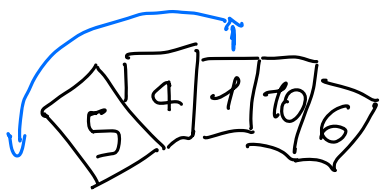
PRIMA DEL main() → #define DIM 6

NEL main() → int vet[DIM] = { 5, 10, 2, 3, 1, 4 };

	0	1	2	3	4	5
vet	5	10	2	3	1	4

IDEA: SCORRO IL VETTORE DA SINISTRA A DESTRA E VADO A POSIZIONARE I VALORI NELLA POSIZIONE CORRETTA.

COME ORDINARE LE CARTE IN UNA MANO.



	0	1	2	3	4	5
vet	5	10	2	3	1	4

j i

NON CI SONO SCAMBI.

10 È GIÀ ORDINATO RISPETTO AL 5!

	0	1	2	3	4	5
vet	5	10	2	3	1	4

j i

SCAMBIO →

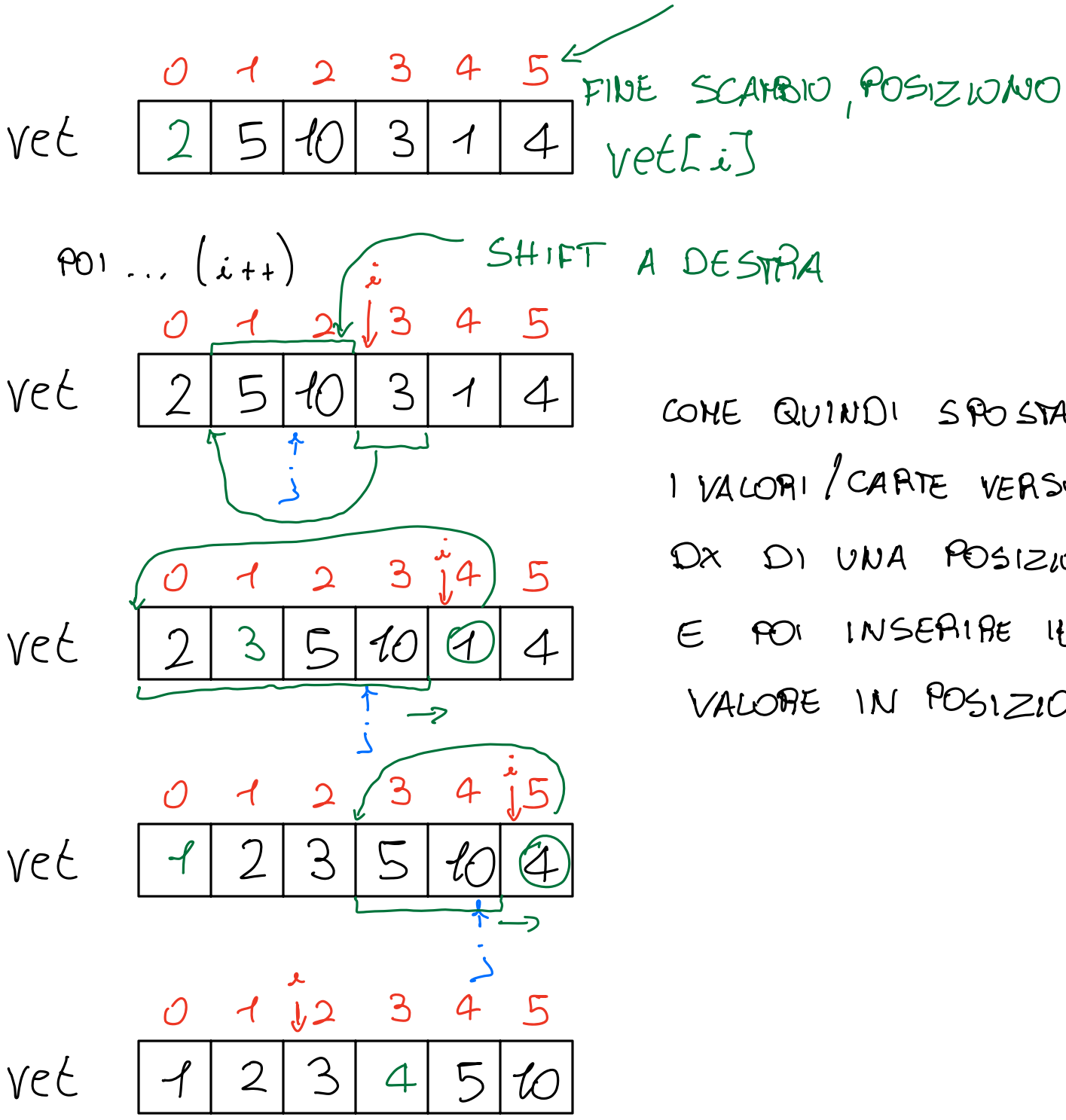
	0	1	2	3	4	5
vet	5	10	10	3	1	4

j i

↓ (j--)

	0	1	2	3	4	5
vet	5	5	10	3	1	4

j i



COME QUINDI SPOSTARE I VALORI / CARTE VERSO DX DI UNA POSIZIONE E POI INSERIRE IL VALORE IN POSIZIONE i.

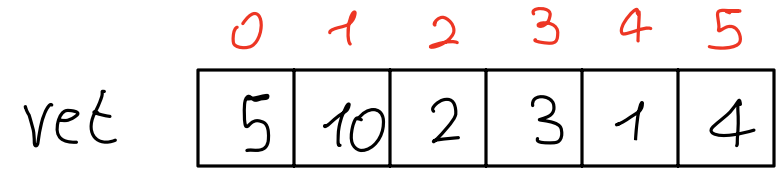
FUNZIONE INSERTION SORT

```
void insertionSort(int vet[], int dim){
    int i, j;
    int key;

    for(i=1; i<dim; i++){
        key = vet[i];
        j = i-1;

        while (j>=0 && vet[j]>key){
            vet[j+1] = vet[j];
            j--;
        }

        vet[j] = key;
    }
}
```

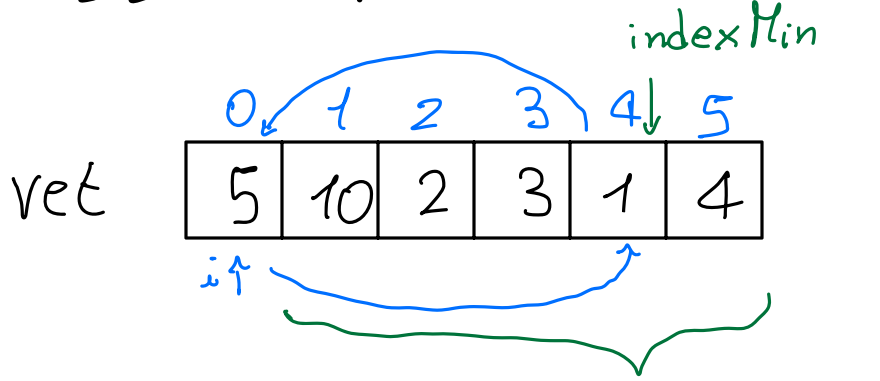
SELECTION SORT

PRIMA DEL main() → #define DIM 6

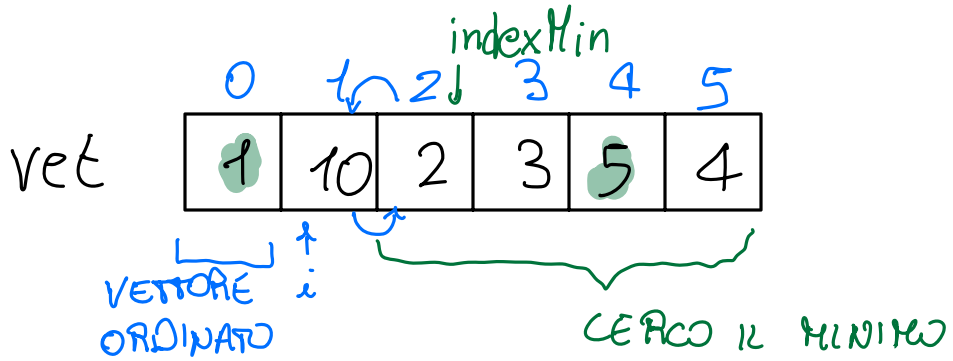
NEL main() → int vet[DIM] = { 5, 10, 2, 3, 1, 4 };

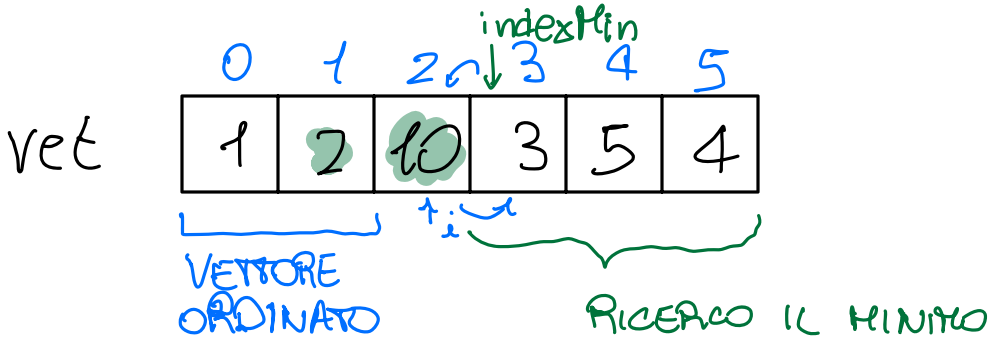


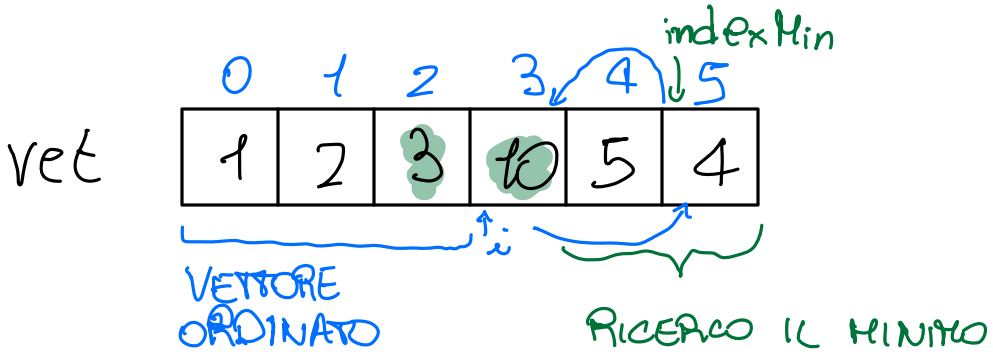
IDEA: TROVO IL MINIMO VALORE DALLA POSIZIONE i FINO ALLA FINE E LO SCAMBIO PER METTERLO NELLA POSIZIONE i.

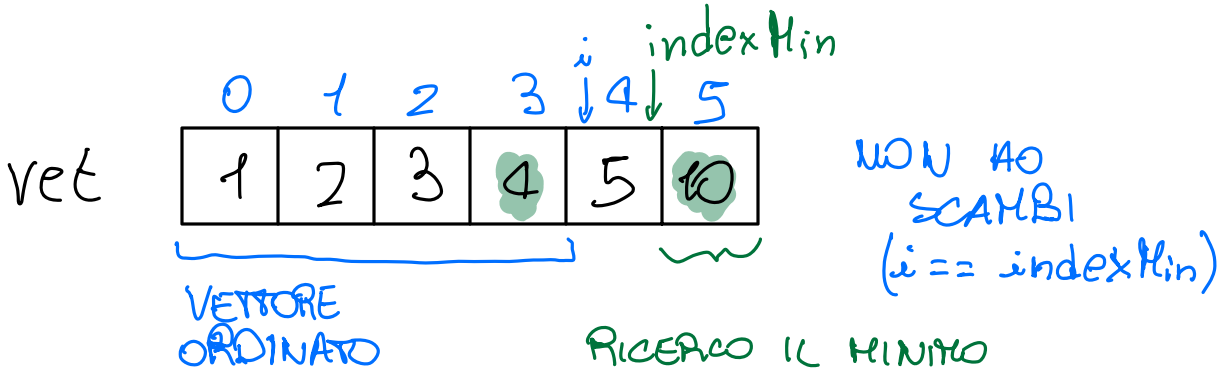


SCAMBIO IL VALORE IN POSIZIONE i CON IL VALORE DI indexMin




indexMin
0 1 2 3 4 5
vet
1 2 10 3 5 4
i
VETTORE ORDINATO
RICERCO IL MINIMO


indexMin
0 1 2 3 4 5
vet
1 2 3 10 5 4
i
VETTORE ORDINATO
RICERCO IL MINIMO


i
indexMin
0 1 2 3 4 5
vet
1 2 3 4 5 10
VETTORE ORDINATO
RICERCO IL MINIMO
NON HO SCAMBI
(i == indexMin)

FUNZIONE SELECTION SORT

```
void selectionSort (int vet[], int dim){
    int i,j;
    int indexMinVal;
    for (i=0; i<dim-1; i++){
        indexMinVal = i;
        for(j=i+1; i<dim; j++){
            if(vet[i] > vet[indexMinVal]){
                indexMinVal = i;
            }
        }
        if (indexMinVal != i){
            swap(&(vet[i]), &(vet[indexMinVal]));
        }
    }
```