# Monitoria Algoritmos

Ps: A sala lota, então quem não for da monitoria, por favor se retire

# Questão do huxley: 56, 77 e 187

Spoj:VARYINGFOCUS

# HeapSort

```
int arr[]={4,5,1,2,3,7,8,6};
```

|4|5|1|2|3|7|8|6|

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
```

|4|5|1|2|3|7|8|6|

```
void buildheap(int size){
        for(int i=(size)/2-1;i>=0;i--)
        siftdown(i);
}
```

|4|5|1|2|3|7|8|6|

```
void buildheap(int size){
        for(int i=(size)/2-1;i>=0;i--)
        siftdown(i);
}
```

|4|5|1|2|3|7|8|6|

i=3

|4|5|1|6|3|7|8|2|

```
void buildheap(int size){
        for(int i=(size)/2-1;i>=0;i--)
        siftdown(i);
}
```

|4|5|1|2|3|7|8|6|

i=2

|4|5|8|6|3|7|1|2|

```
void buildheap(int size){
        for(int i=(size)/2-1;i>=0;i--)
        siftdown(i);
}
```

|4|5|1|2|3|7|8|6|

i=1

|4|6|8|5|3|7|1|2|

```
void buildheap(int size){
        for(int i=(size)/2-1;i>=0;i--)
        siftdown(i);
}
```

|4|5|1|2|3|7|8|6|

i=0

|8|6|7|5|3|4|1|2|

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
```

|4|5|1|2|3|7|8|6|

|8|6|7|5|3|4|1|2|

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2| temp=8 size=8

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2| temp=8 size=8
|7|6|4|5|3|2|1|2| temp=8 size=7

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2| temp=8 size=8
|7|6|4|5|3|2|1|2| temp=8 size=7
|7|6|4|5|3|2|1|8| temp=8 size=7

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|7|6|4|5|3|2|1|8| temp=7 size=7

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|7|6|4|5|3|2|1|8| temp=7 size=7
|6|5|4|1|3|2|1|8| temp=7 size=6

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|7|6|4|5|3|2|1|8| temp=7 size=7
|6|5|4|1|3|2|1|8| temp=7 size=6
|6|5|4|1|3|2|7|8| temp=7 size=6

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|6|5|4|1|3|2|7|8| temp=6 size=6

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|6|5|4|1|3|2|7|8| temp=6 size=6
|5|3|4|1|2|2|7|8| temp=6 size=5

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|6|5|4|1|3|2|7|8| temp=6 size=6
|5|3|4|1|2|2|7|8| temp=6 size=5
|5|3|4|1|2|6|7|8| temp=6 size=5

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|5|3|4|1|2|6|7|8| temp=5 size=5

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|5|3|4|1|2|6|7|8| temp=5 size=5
|4|3|2|1|2|6|7|8| temp=5 size=4

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|5|3|4|1|2|6|7|8| temp=5 size=5
|4|3|2|1|2|6|7|8| temp=5 size=4
|4|3|2|1|5|6|7|8| temp=5 size=4

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|4|3|2|1|5|6|7|8| temp=4 size=4

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|4|3|2|1|5|6|7|8| temp=4 size=4
|3|1|2|1|5|6|7|8| temp=4 size=3

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|4|3|2|1|5|6|7|8| temp=4 size=4
|3|1|2|1|5|6|7|8| temp=4 size=3
|3|1|2|4|5|6|7|8| temp=4 size=3

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|3|1|2|4|5|6|7|8| temp=3 size=3

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|3|1|2|4|5|6|7|8| temp=3 size=3
|2|1|2|4|5|6|7|8| temp=3 size=2

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|3|1|2|4|5|6|7|8| temp=3 size=3
|2|1|2|4|5|6|7|8| temp=3 size=2
|2|1|3|4|5|6|7|8| temp=3 size=2

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|2|1|3|4|5|6|7|8| temp=2 size=2

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|2|1|3|4|5|6|7|8| temp=2 size=2
|1|1|3|4|5|6|7|8| temp=2 size=1

```
int arr[]={4,5,1,2,3,7,8,6};
buildMaxHeap();//iniciando em Zero
while(size>1){
        int temp=arr[0];
        pop(&arr);
        arr[size]=temp;
}
```

|4|5|1|2|3|7|8|6|
|8|6|7|5|3|4|1|2|
|8|6|7|5|3|4|1|2|
|2|1|3|4|5|6|7|8| temp=2 size=2
|1|1|3|4|5|6|7|8| temp=2 size=1
|1|2|3|4|5|6|7|8| temp=2 size=1

# MergeSort

```
int main(){

int a[] = {5,3,4,7,1,8,2,6};

sort(a, 0, size-1);

return 0;

}
```

| 5 | 3 | 4 | 7 | 1 | 8 | 2 | 6 |
|---|---|---|---|---|---|---|---|

```
void sort(int *a, int ini, int end){

if(ini == end) return;

sort(a, ini, (ini+end)/2);

sort(a, ((ini+end)/2) +1, end);

merge(a, ini, ((ini+end)/2) +1, end);

}
```

| 5 | 3 | 4 | 7 | 1 | 8 | 2 | 6 |
|---|---|---|---|---|---|---|---|

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

```
void sort(int *a, int ini, int end){

if(ini == end) return;

sort(a, ini, (ini+end)/2);

sort(a, ((ini+end)/2) +1, end);

merge(a, ini, ((ini+end)/2) +1, end);

}
```

retorna

end
ini

| 5 | 3 | 4 | 7 | 1 | 8 | 2 | 6 |
|---|---|---|---|---|---|---|---|

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

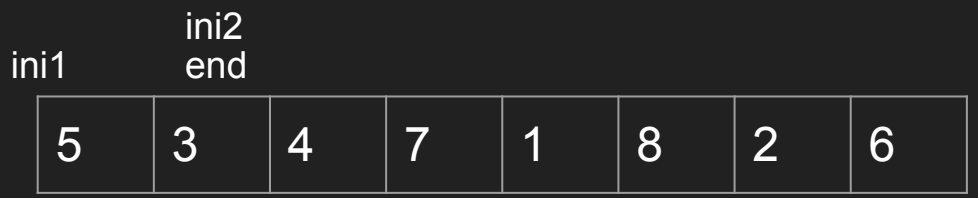
```
void sort(int *a, int ini, int end){

if(ini == end) return;

sort(a, ini, (ini+end)/2);

sort(a, ((ini+end)/2) +1, end);

merge(a, ini, ((ini+end)/2) +1, end);

}
```

retorna

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```



```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
    while(i < ini2){ b[k] = a[i]; i++; k++;}
    while(j < end+1){ b[k] = a[j]; j++; k++;}
     k = 0;
      i = ini1;
     while(i <= end) {
            a[i] = b[k];
           i++;  k++;
     }
```

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
    while(i < ini2){ b[k] = a[i]; i++; k++;}
    while(j < end+1){ b[k] = a[j]; j++; k++;}
     k = 0;
      i = ini1;
     while(i <= end) {
            a[i] = b[k];
           i++;  k++;
     }
```

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```

b

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;
        }
    }
    while(i < ini2){ b[k] = a[i]; i++; k++;}
    while(j < end+1){ b[k] = a[j]; j++; k++;}
    k = 0;
    i = ini1;
    while(i <= end) {
        a[i] = b[k];
        i++;  k++;
    }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }   }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```

b

k

ini1 ini2 end


a

i j

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }   }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```

b

k

ini1 ini2 end

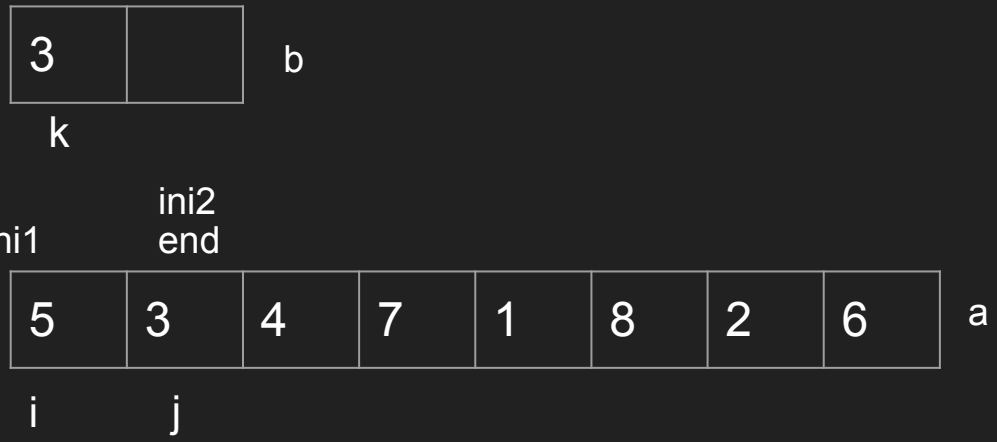

a

i j

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;
        }
    }
    while(i < ini2){ b[k] = a[i]; i++; k++;}
    while(j < end+1){ b[k] = a[j]; j++; k++;}
    k = 0;
    i = ini1;
    while(i <= end) {
        a[i] = b[k];
        i++;  k++;
    }
```

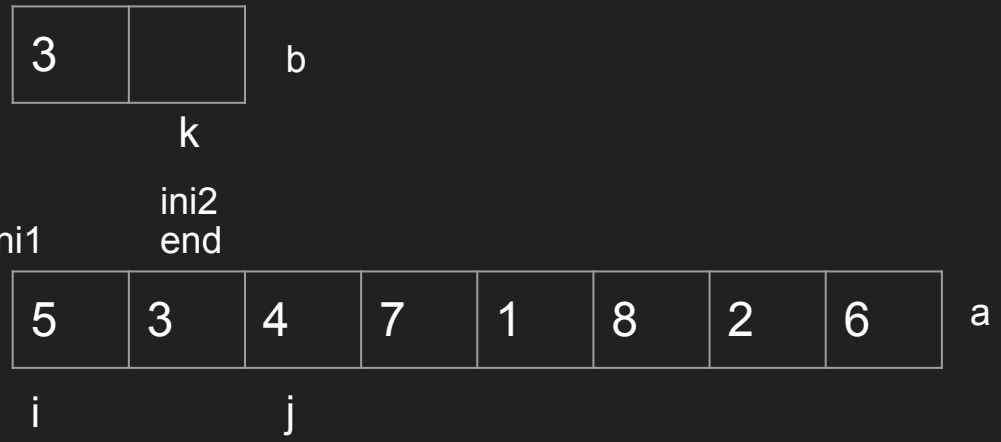

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```



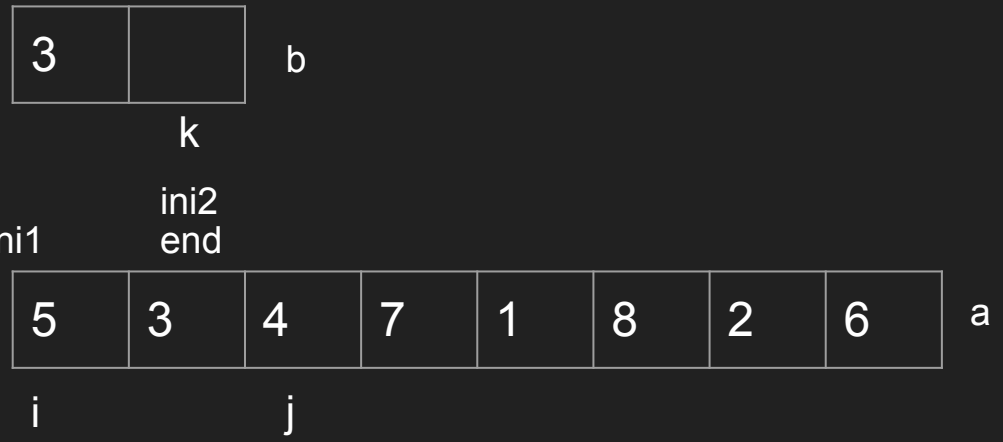
```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```



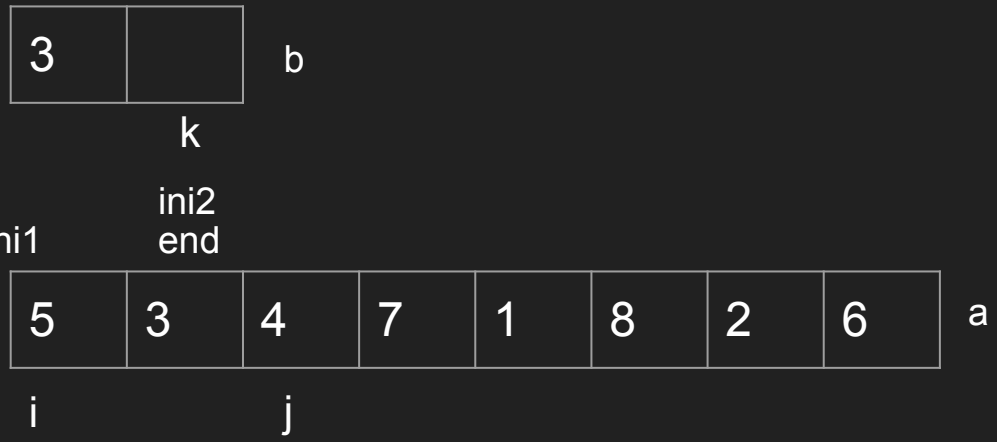
```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```



```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```

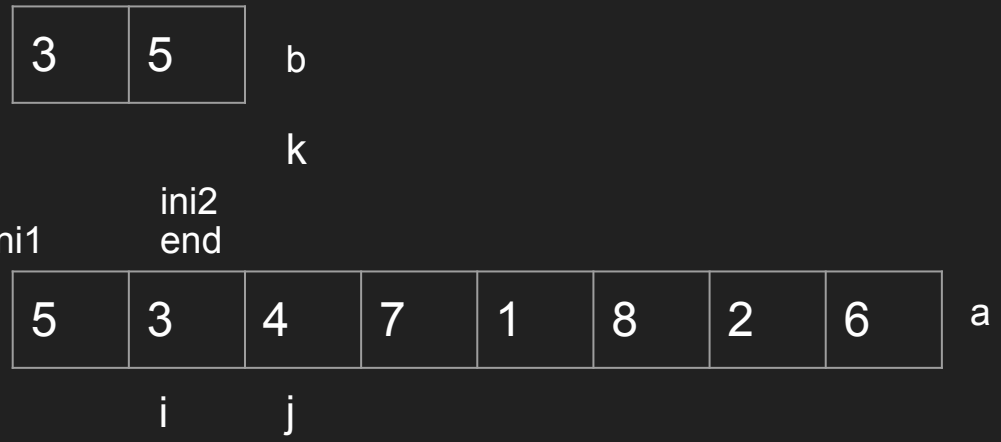


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```

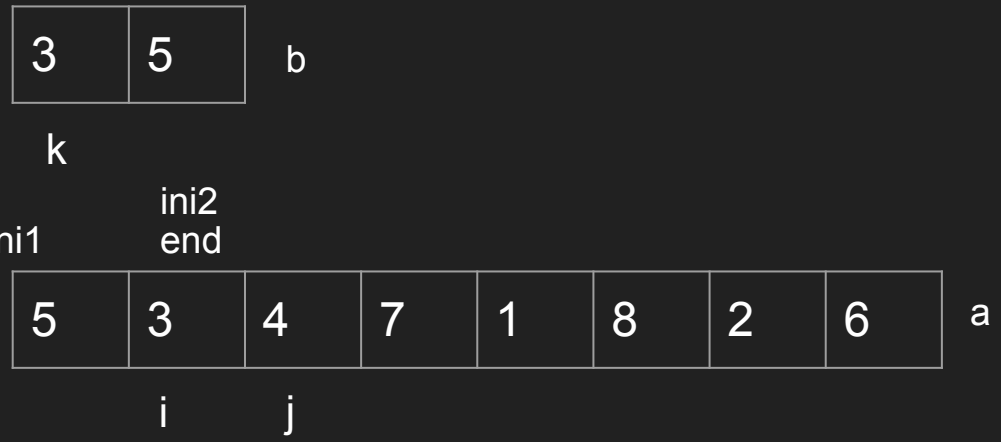
```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```



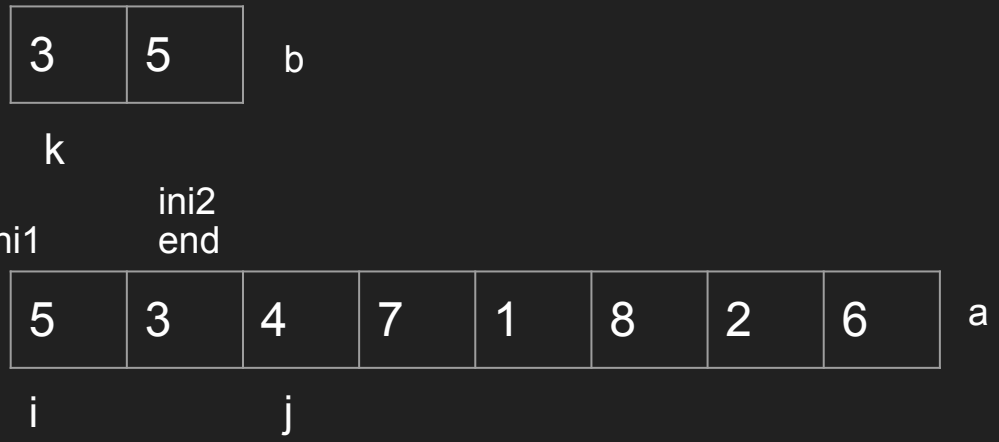
```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```



```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```



```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
           j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


b

k

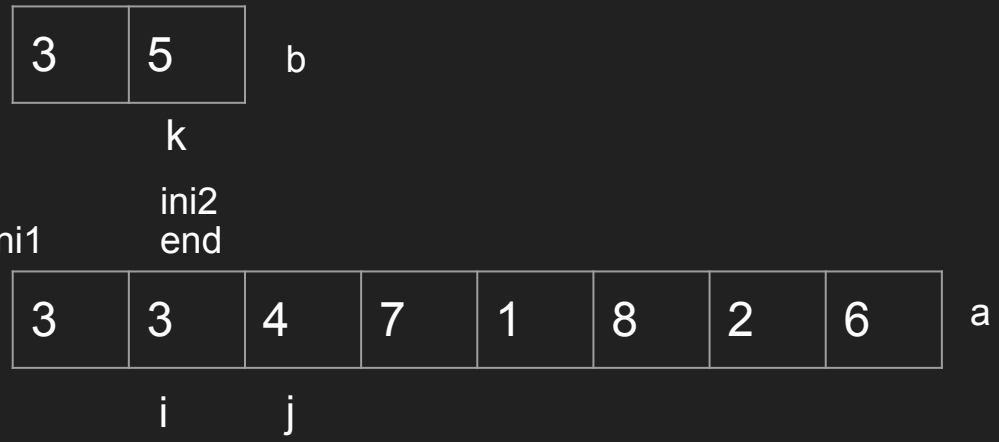
ini1 ini2 end


a

i j

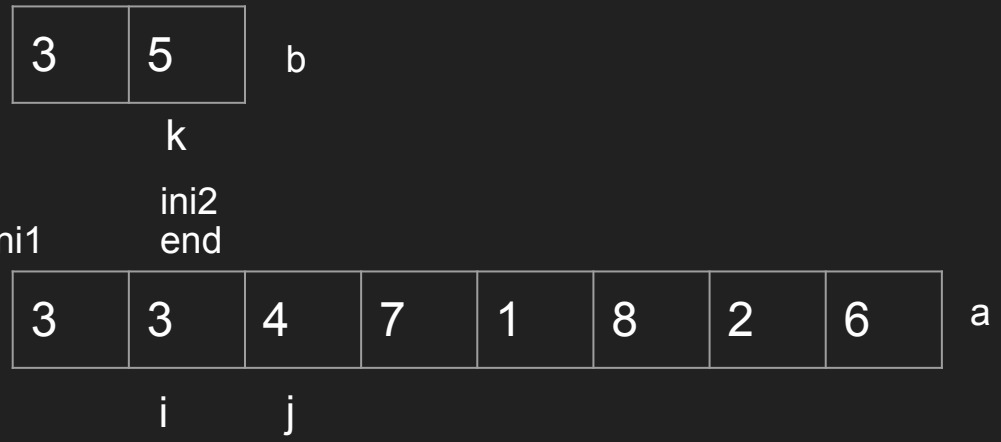
```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```



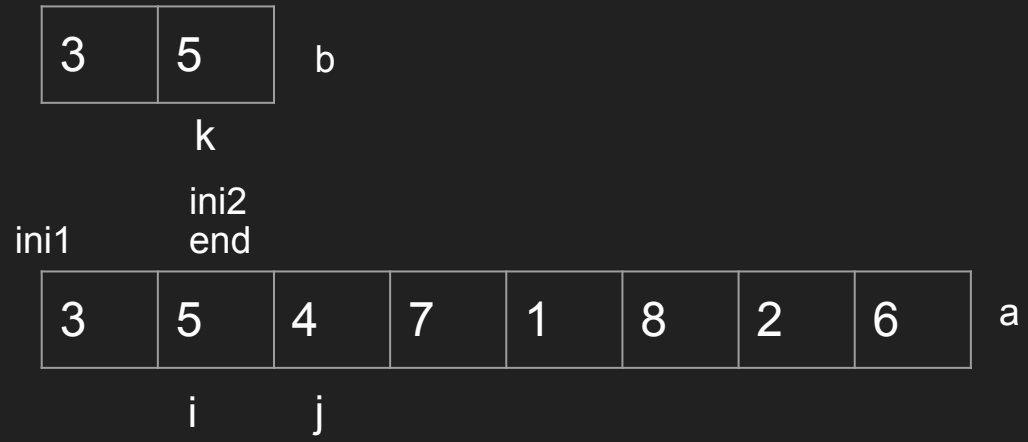
```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```



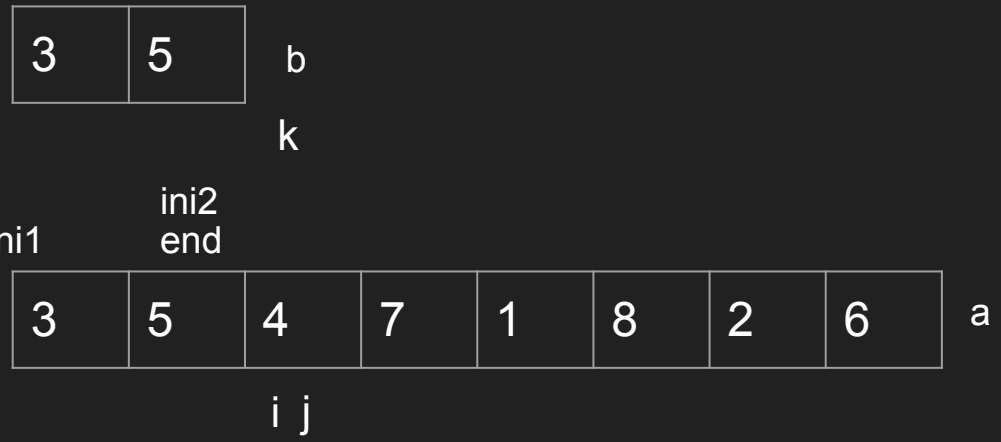
```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
        i++;  k++;
 }
```



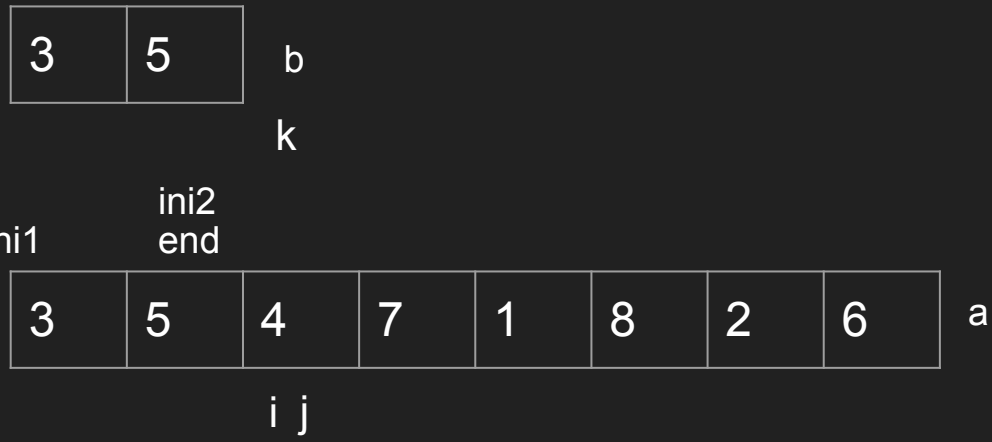
```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
        i++;  k++;
 }
```



```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```

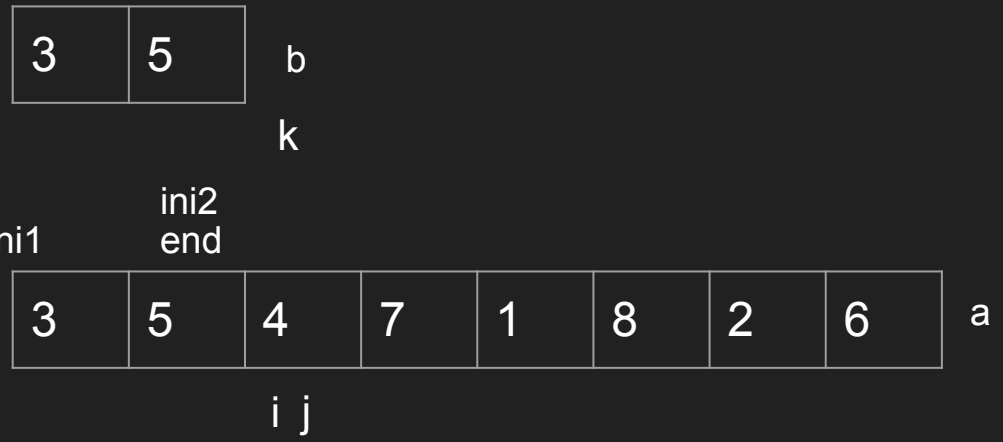


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```



```
void sort(int *a, int ini, int end){

if(ini == end) return;

sort(a, ini, (ini+end)/2);

sort(a, ((ini+end)/2) +1, end);

merge(a, ini, ((ini+end)/2) +1, end);

}
```

```
void sort(int *a, int ini, int end){

if(ini == end) return;

sort(a, ini, (ini+end)/2);

sort(a, ((ini+end)/2) +1, end);

merge(a, ini, ((ini+end)/2) +1, end);

}
```

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

ini end

| 3 | 5 | 4 | 7 | 1 | 8 | 2 | 6 |
|---|---|---|---|---|---|---|---|

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

```
void sort(int *a, int ini, int end){

if(ini == end) return;

sort(a, ini, (ini+end)/2);

sort(a, ((ini+end)/2) +1, end);

merge(a, ini, ((ini+end)/2) +1, end);

}
```

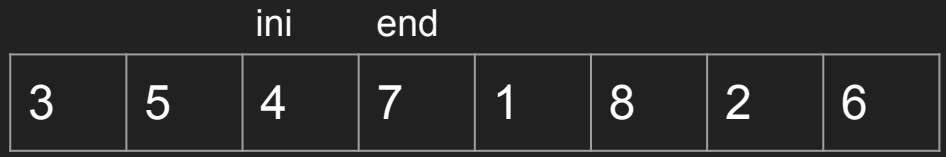
retorna

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

retorna

```
void sort(int *a, int ini, int end){

if(ini == end) return;

sort(a, ini, (ini+end)/2);

sort(a, ((ini+end)/2) +1, end);

merge(a, ini, ((ini+end)/2) +1, end);

}
```




Já vimos o merge funcionando para dois elementos :)
pularemos para o próximo passo

```
void sort(int *a, int ini, int end){
if(ini == end) return;
sort(a, ini, (ini+end)/2);
sort(a, ((ini+end)/2) +1, end);
merge(a, ini, ((ini+end)/2) +1, end);
}
```

```
void sort(int *a, int ini, int end){

if(ini == end) return;

sort(a, ini, (ini+end)/2);

sort(a, ((ini+end)/2) +1, end);

merge(a, ini, ((ini+end)/2) +1, end);

}
```

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
    while(i < ini2){ b[k] = a[i]; i++; k++;}
    while(j < end+1){ b[k] = a[j]; j++; k++;}
     k = 0;
      i = ini1;
     while(i <= end) {
            a[i] = b[k];
           i++;  k++;
     }
```

| ini1 | | ini2 | end | | | | |
|---|---|---|---|---|---|---|---|
| 3 | 5 | 4 | 7 | 1 | 8 | 2 | 6 |

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
    while(i < ini2){ b[k] = a[i]; i++; k++;}
    while(j < end+1){ b[k] = a[j]; j++; k++;}
     k = 0;
      i = ini1;
     while(i <= end) {
            a[i] = b[k];
           i++;  k++;
     }
```

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
    while(i < ini2){ b[k] = a[i]; i++; k++;}
    while(j < end+1){ b[k] = a[j]; j++; k++;}
     k = 0;
      i = ini1;
     while(i <= end) {
            a[i] = b[k];
           i++;  k++;
     }
```

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```

b

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;
        }
    }
    while(i < ini2){ b[k] = a[i]; i++; k++;}
    while(j < end+1){ b[k] = a[j]; j++; k++;}
    k = 0;
    i = ini1;
    while(i <= end) {
        a[i] = b[k];
        i++;  k++;
    }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;
        }
    }
    while(i < ini2){ b[k] = a[i]; i++; k++;}
    while(j < end+1){ b[k] = a[j]; j++; k++;}
    k = 0;
    i = ini1;
    while(i <= end) {
        a[i] = b[k];
        i++;  k++;
    }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;
        }
    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
    a[i] = b[k];
    i++;  k++;
}
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }   }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```

| 3 | 4 | | | b |
|---|---|---|---|---|

k

ini1 ini2 end

| 3 | 5 | 4 | 7 | 1 | 8 | 2 | 6 | a |
|---|---|---|---|---|---|---|---|---|

i j

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```

| 3 | 4 | 5 | | b |
|---|---|---|---|---|

k

ini1 ini2 end

| 3 | 5 | 4 | 7 | 1 | 8 | 2 | 6 | a |
|---|---|---|---|---|---|---|---|---|

i j

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```

| 3 | 4 | 5 | | b |
|---|---|---|---|---|

k

ini1 ini2 end

| 3 | 5 | 4 | 7 | 1 | 8 | 2 | 6 | a |
|---|---|---|---|---|---|---|---|---|

i j

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;
        }
    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
    a[i] = b[k];
    i++;  k++;
}
```

| 3 | 4 | 5 | | b |
|---|---|---|---|---|

k

ini1 ini2 end

| 3 | 5 | 4 | 7 | 1 | 8 | 2 | 6 | a |
|---|---|---|---|---|---|---|---|---|

i j

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```

| 3 | 4 | 5 | | b |
|---|---|---|---|---|

k

ini1 ini2 end

| 3 | 5 | 4 | 7 | 1 | 8 | 2 | 6 | a |
|---|---|---|---|---|---|---|---|---|

i j

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```


b

k

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```


b

k

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```


b

k

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
        i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
        i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

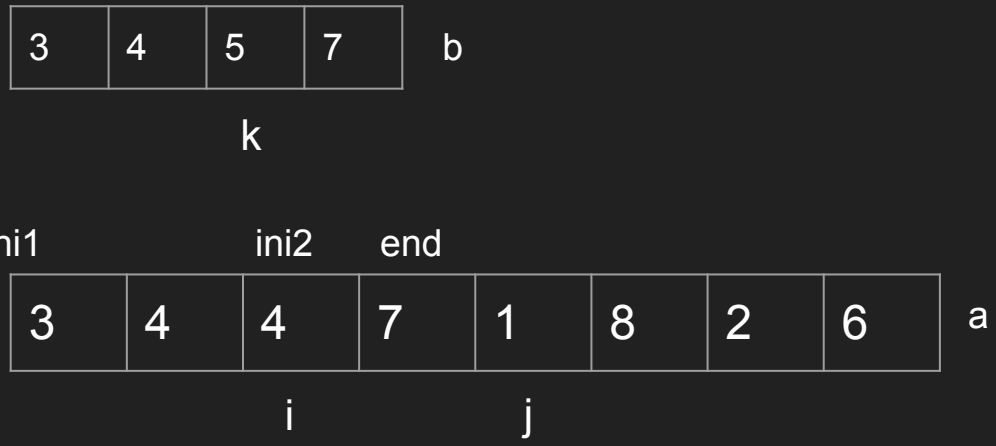
```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
        i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
           j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
        i++;  k++;
 }
```



```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

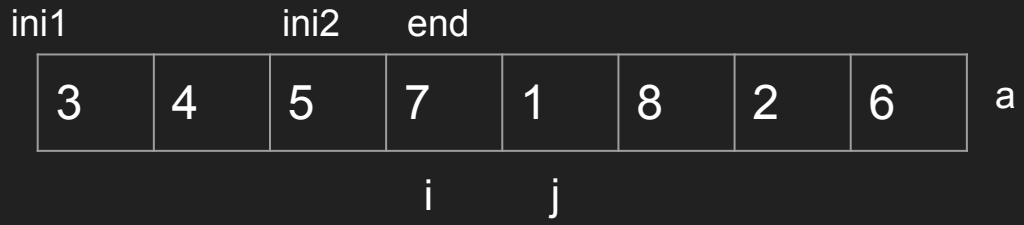
```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
           j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
        i++;  k++;
 }
```

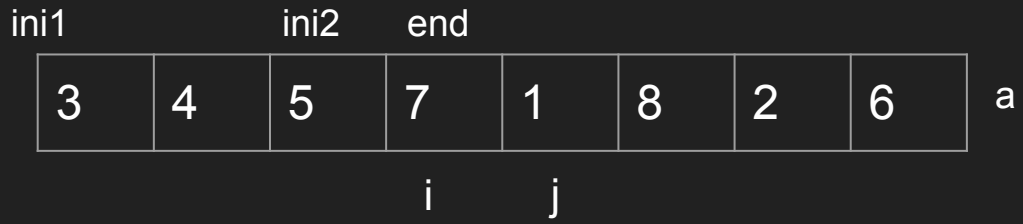



```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```


```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
        i++;  k++;
 }
```

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
        i++;  k++;
 }
```


b

k

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
        i++;  k++;
 }
```

| 3 | 4 | 5 | 7 |
|---|---|---|---|

b

k

ini1 ini2 end

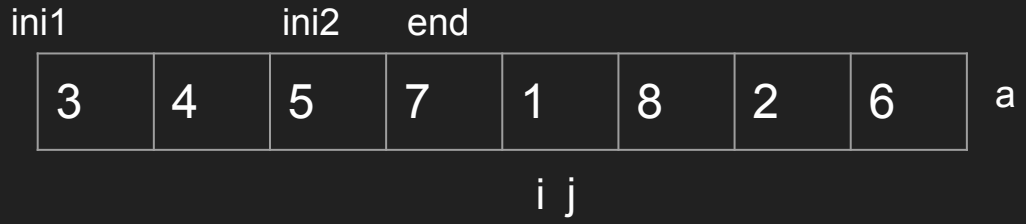
| 3 | 4 | 5 | 7 | 1 | 8 | 2 | 6 |
|---|---|---|---|---|---|---|---|

a

i j

```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;                    }
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
  i = ini1;
 while(i <= end) {
        a[i] = b[k];
       i++;  k++;
 }
```

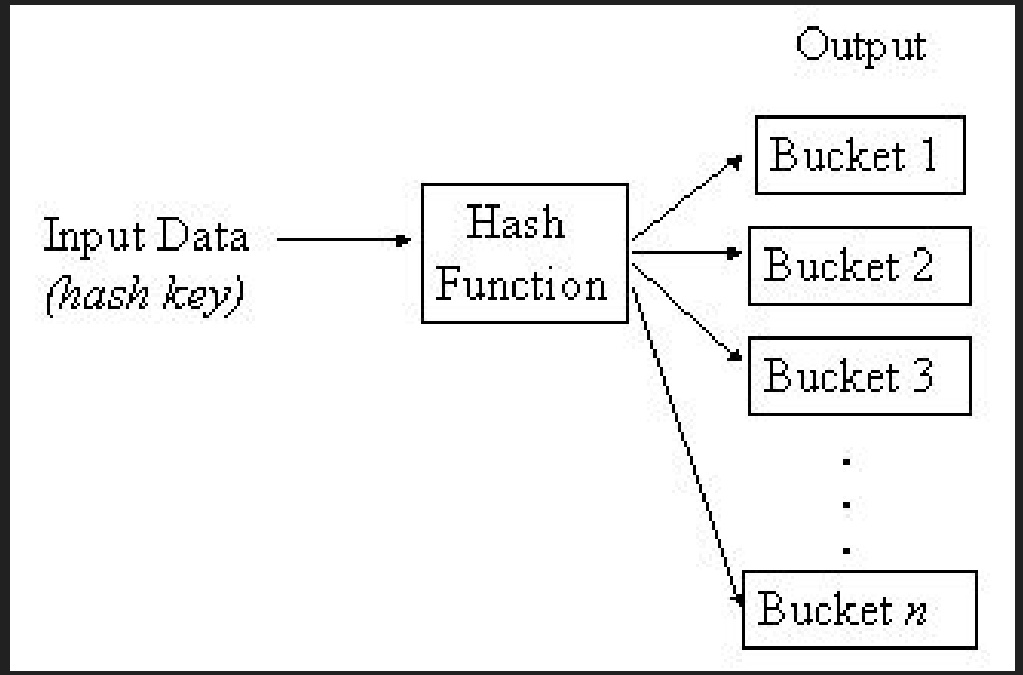
```
void merge(int *a, int ini1, ini2, int end){
    int i = ini1;
    int j = ini2;
    int b[end-i+1];
    int k = 0;
    while(i < ini2 && j < (end+1)){
        if(a[i] < a[j]){
            b[k] = a[i];
            i++; k++;
        }
        else {
            b[k] = a[j];
            j++; k++;   }    }
```

```
while(i < ini2){ b[k] = a[i]; i++; k++;}
while(j < end+1){ b[k] = a[j]; j++; k++;}
 k = 0;
 i = ini1;
while(i <= end) {
        a[i] = b[k];
        i++;  k++;
}
```


Agora que já vimos como o código funciona, vamos observar como fica a tabela do mergesort por completa!!

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 7 | 1 | 8 | 2 | 6 |
| 5 | 3 | 4 | 7 | 1 | 8 | 2 | 6 |
| 5 | 3 | 4 | 7 | 1 | 8 | 2 | 6 |
| 3 | 5 | 4 | 7 | 1 | 8 | 2 | 6 |
| 3 | 5 | 4 | 7 | 1 | 8 | 2 | 6 |
| 3 | 4 | 5 | 7 | 1 | 8 | 2 | 6 |
| 3 | 4 | 5 | 7 | 1 | 8 | 2 | 6 |
| 3 | 4 | 5 | 7 | 1 | 8 | 2 | 6 |
| 3 | 4 | 5 | 7 | 1 | 8 | 2 | 6 |
| 3 | 4 | 5 | 7 | 1 | 2 | 6 | 8 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

# Hashing


Output
Input Data
(hash key)
Hash
Function
Bucket 1
Bucket 2
Bucket 3
Bucket n

```c
#define TAM 3

typedef struct Pessoa
{
    int id;
    char *nome;
} Pessoa;


typedef struct Tabela
{
    Pessoa *p;
    struct Tabela *next;
} Tabela;

Tabela **t;
```

```c
int hash(int ano)
{
    return ano % TAM;
}

void inicializar()
{
    t = (Tabela**)malloc(sizeof(Tabela*)*TAM);

    int i;
    for (i = 0; i < TAM; ++i)
    {
        t[i] = NULL;
    }
}
```

```c
void inserir(Pessoa *pessoa)
{
    int pos = hash(pessoa->id);


    if (t[pos] == NULL)
    {
        //caso lista vazia
        t[pos] = (Tabela*)malloc(sizeof(Tabela));
        t[pos]->p = pessoa;
        t[pos]->next = NULL;
    }
    else
    {
        //adicionando no fim da lista
        Tabela *taux = t[pos];
        while (taux->next != NULL)
        {
            taux = taux->next;
        }
        taux->next = (Tabela*)malloc(sizeof(Tabela));
        taux->next->p = pessoa;
        taux->next->next = NULL;
    }
}
```

```
63  Pessoa* procurar(int id)
64  {
65      int pos = hash(id);
66      Tabela *aux = t[pos];
67
68      while (aux != NULL)
69      {
70          if (aux->p->id == id)
71          {
72              return aux->p;
73          }
74          aux = aux->next;
75      }
76
77      return NULL;
78  }
79
```

```
80  void limpaNo(Tabela *taux)
81  {
82      if (taux == NULL) return;
83      limpaNo(taux->next);
84      free(taux);
85  }
86
87  void limpar()
88  {
89      int i;
90      for (i = 0; i < TAM; ++i)
91      {
92          limpaNo(t[i]);
93      }
94  }
95
96  void limpaPessoa(Pessoa **p, int size)
97  {
98      int i;
99      for (i = 0; i < size; ++i)
100     {
101         free(p[i]);
102     }
103
104     free(p);
105 }
```

```c
int main()
{
    inicializar();
    int size = 5;
    Pessoa **p = (Pessoa**)malloc(sizeof(Pessoa*)*size);

    int i;
    for (i = 0; i < size; ++i)
    {
        p[i] = (Pessoa*)malloc(sizeof(Pessoa));
    }

    p[0]->id = 600;
    p[1]->id = 100;
    p[2]->id = 3;
    p[3]->id = 6;
    p[4]->id = 1;

    p[0]->nome = "katia";
    p[1]->nome = "Carlos";
    p[2]->nome =  "Allyson";
    p[3]->nome = "Arthur";
    p[4]->nome = "Jonatas";

    for (i = 0; i < size; ++i)
    {
        inserir(p[i]);
    }

    Pessoa *a = procurar(1);

    limpar();

    limpaPessoa(p, size);
    return 0;
```