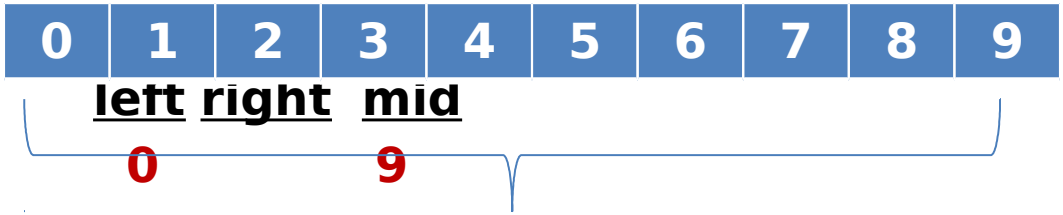
Searching in Arrays

Recursive Binary Search

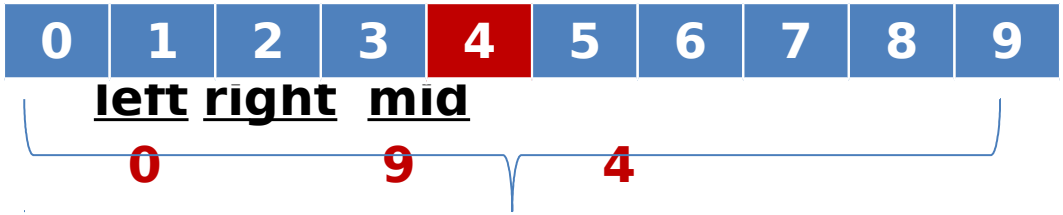
```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in
left side
    return binHelper (array, key, mid+1, right);            //seach in
right side
}

public static int binSearch(int[]array, int key)
{
    return binHelper(array, key, 0, array.length -1);
}
```

```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in
left side
    return binHelper (array, key, mid+1, right);            //seach in
right side
}
```



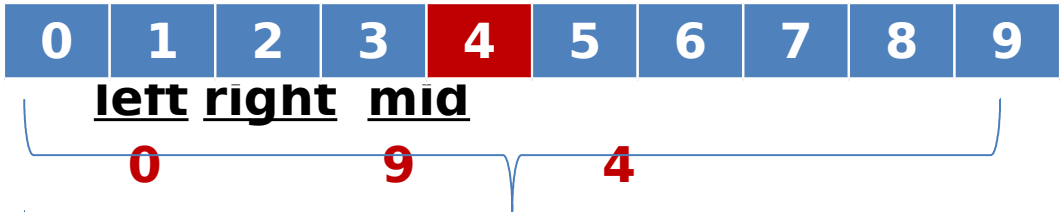
key == 5

```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in
left side
    return binHelper (array, key, mid+1, right);            //seach in
right side
}
```



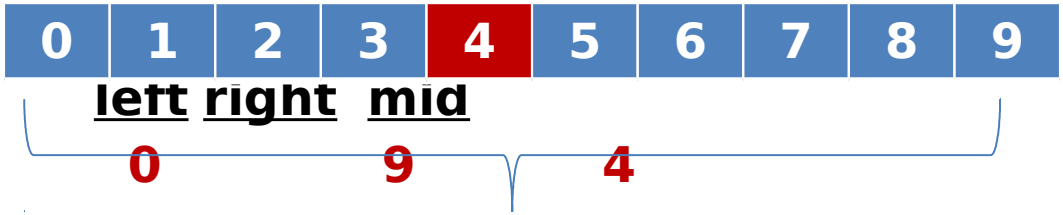
key == 5

```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in
left side
    return binHelper (array, key, mid+1, right);            //seach in
right side
}
```



key == 5

```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in
left side
    return binHelper (array, key, mid+1, right);            //seach in
right side
}
```



key == 5

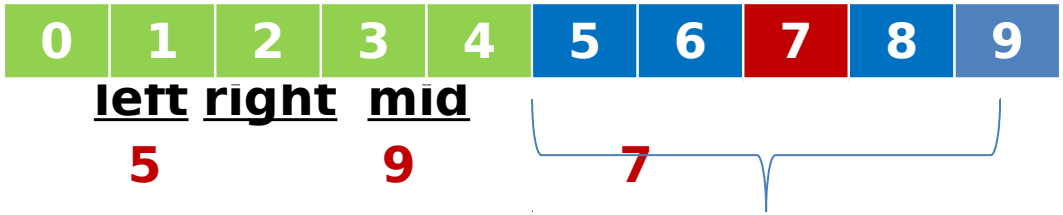
```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in
left side
    return binHelper (array, key, mid+1, right);            //seach in
right side
}
```

key == 5

```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in left side
    return binHelper (array, key, mid+1, right);            //seach in right side
}
```

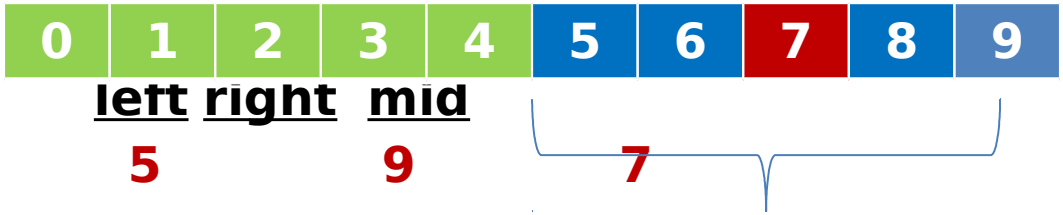
key == 5

```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in
left side
    return binHelper (array, key, mid+1, right);            //seach in
right side
}
```



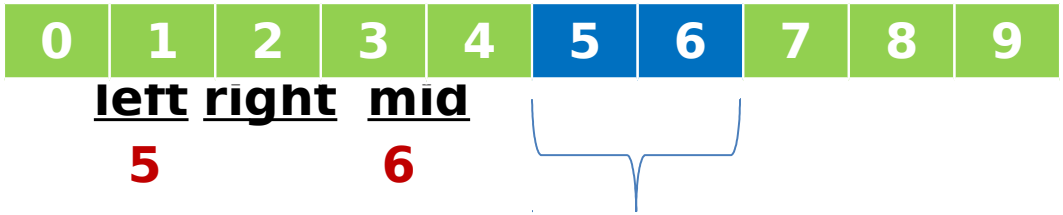
key == 5

```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in
left side
    return binHelper (array, key, mid+1, right);            //seach in
right side
}
```



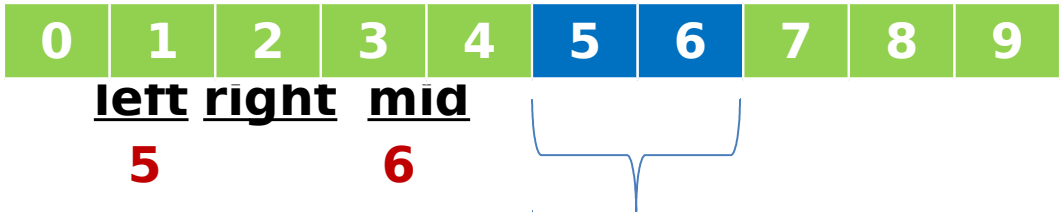
key == 5

```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in
left side
    return binHelper (array, key, mid+1, right);            //seach in
right side
}
```



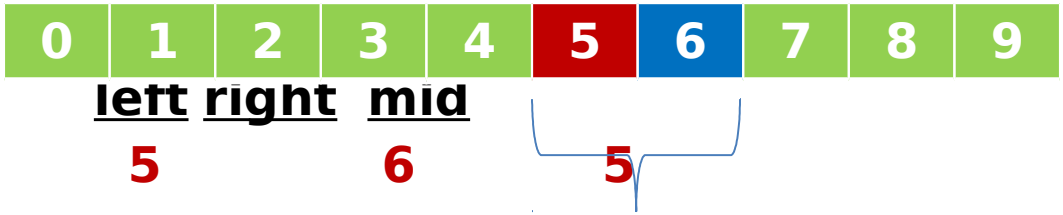
key == 5

```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in left side
    return binHelper (array, key, mid+1, right);            //seach in right side
}
```



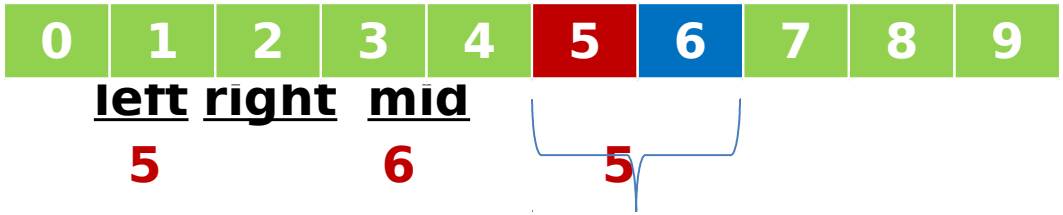
key == 5

```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in
left side
    return binHelper (array, key, mid+1, right);            //seach in
right side
}
```

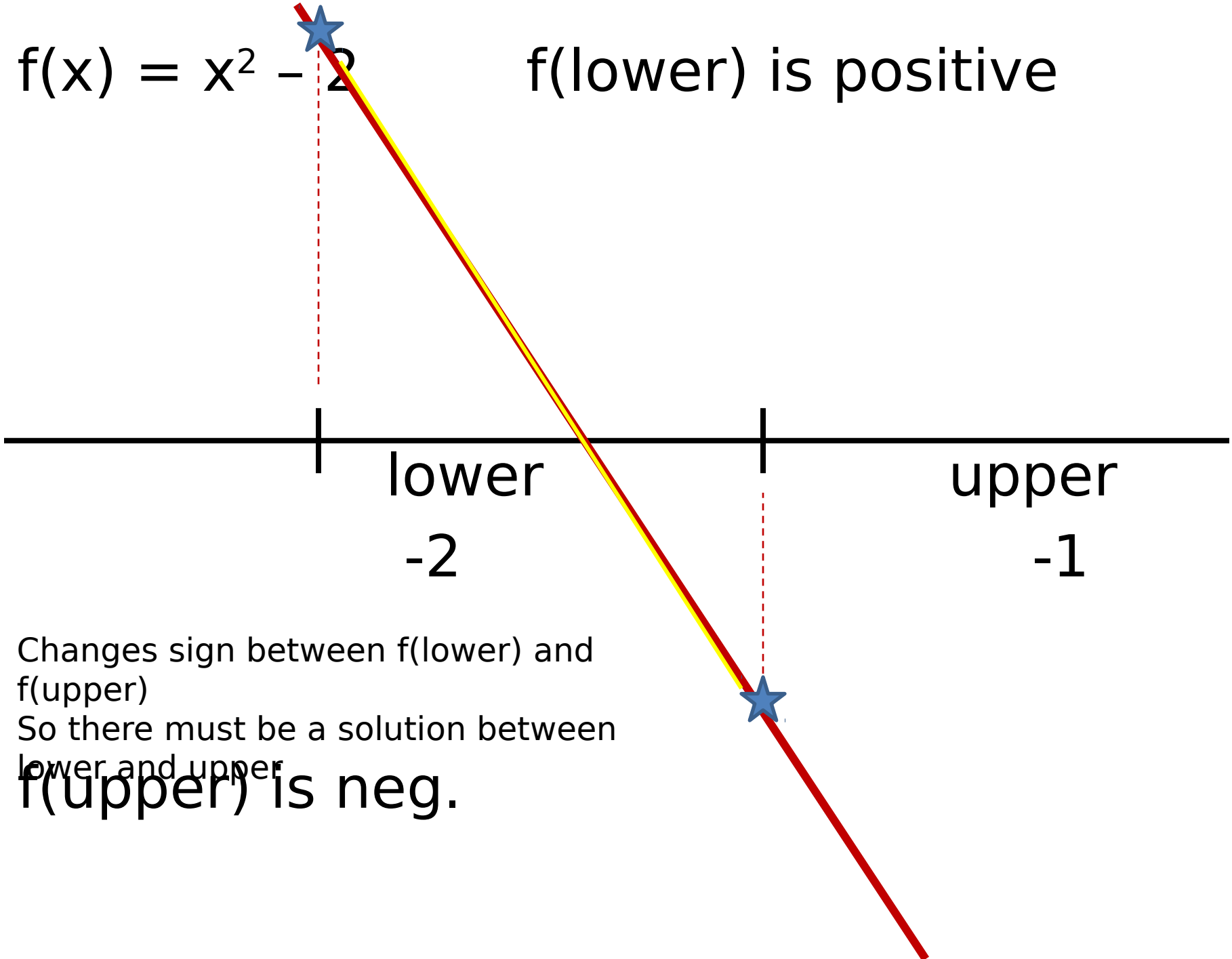


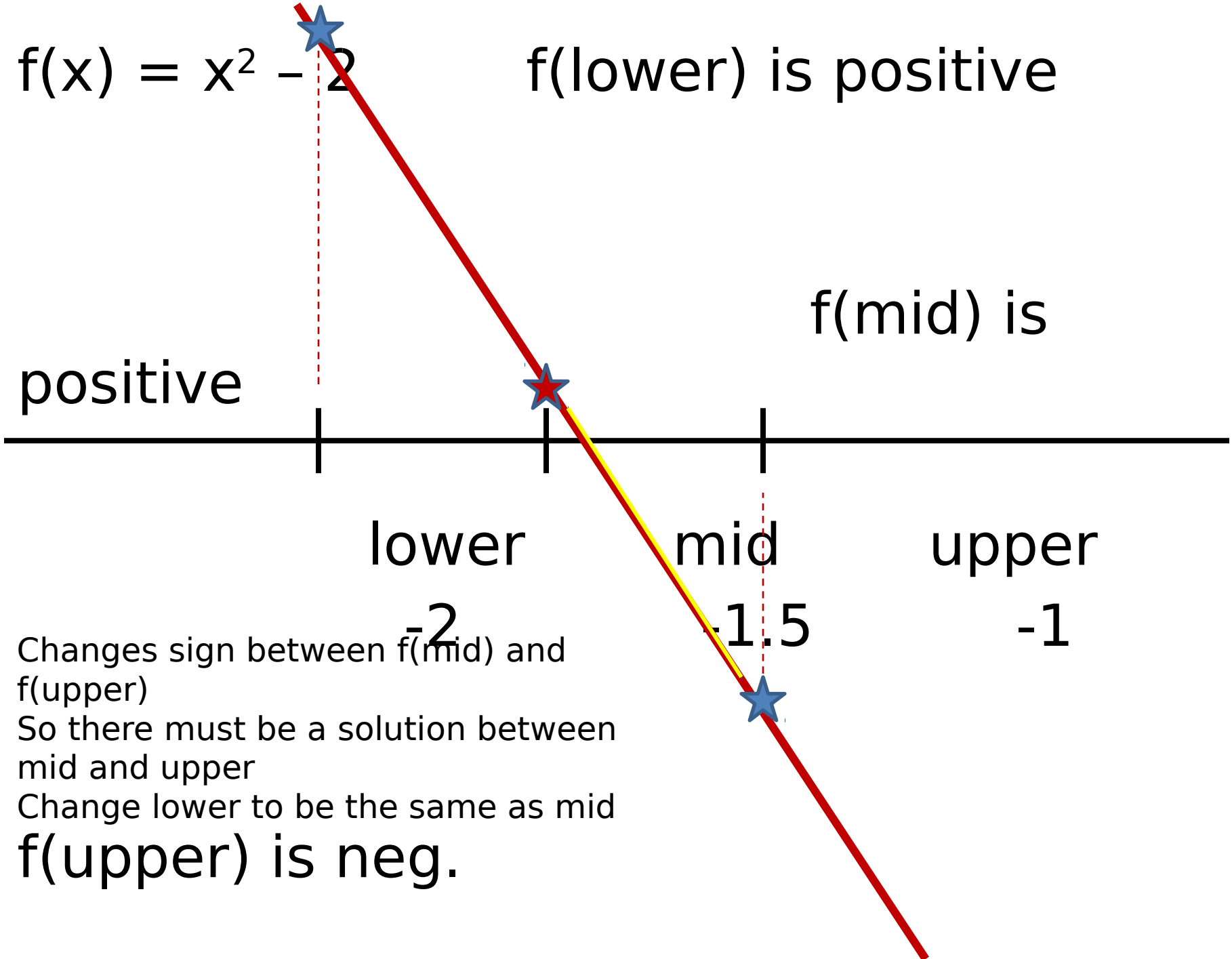
key == 5

```
private static int binHelper(int[]array, int key, int left, int right)
{
    if(left > right)
        return -1;   //key not found in array
    int mid = (left+right)/2;  //find index in the middle of subarray
    if(array[mid] == key)
        return mid;      //we found it – return its index
    if(key < array[mid])
        return binHelper (array, key, left, mid – 1);       //search in
left side
    return binHelper (array, key, mid+1, right);            //seach in
right side
}
```



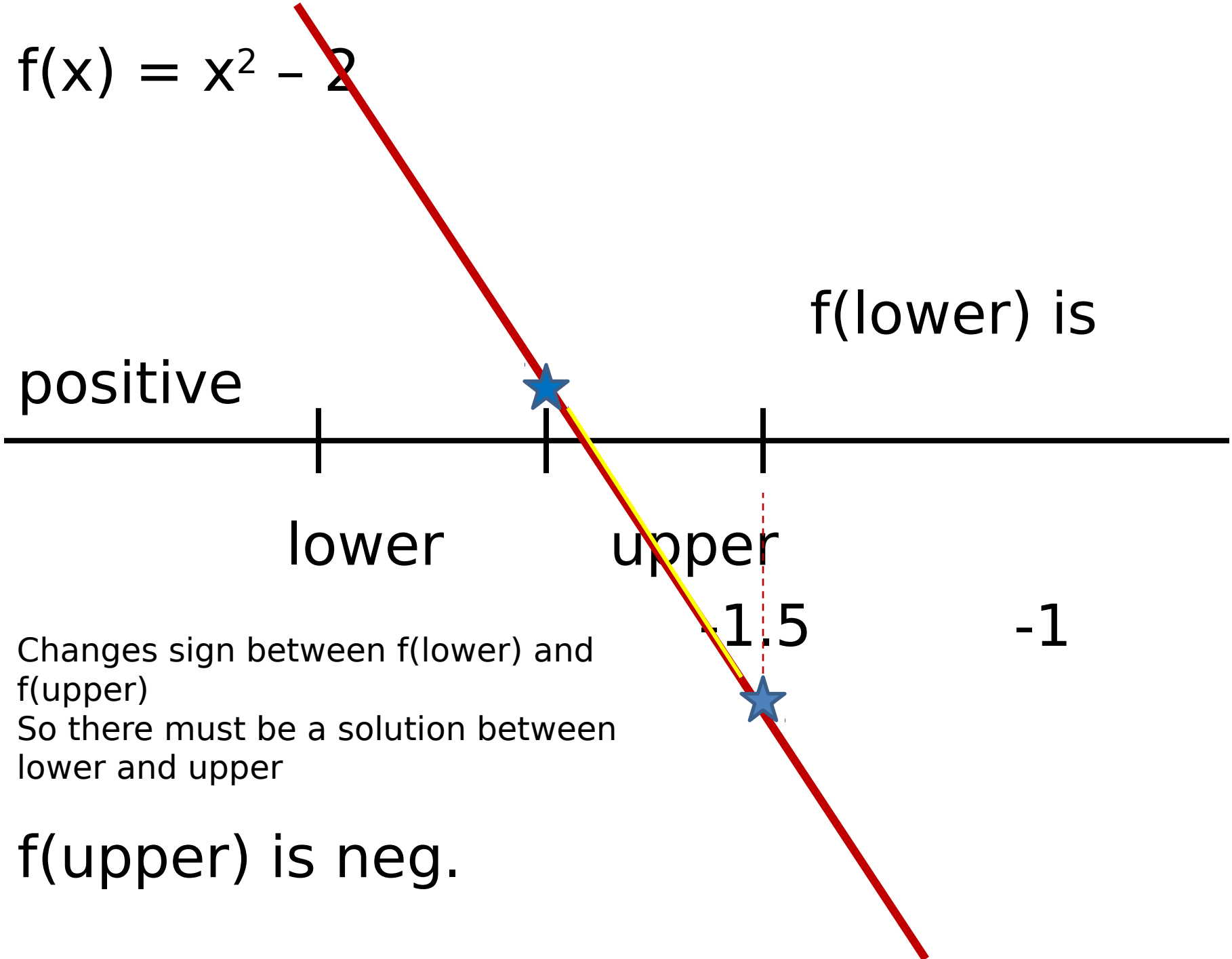
key == 5

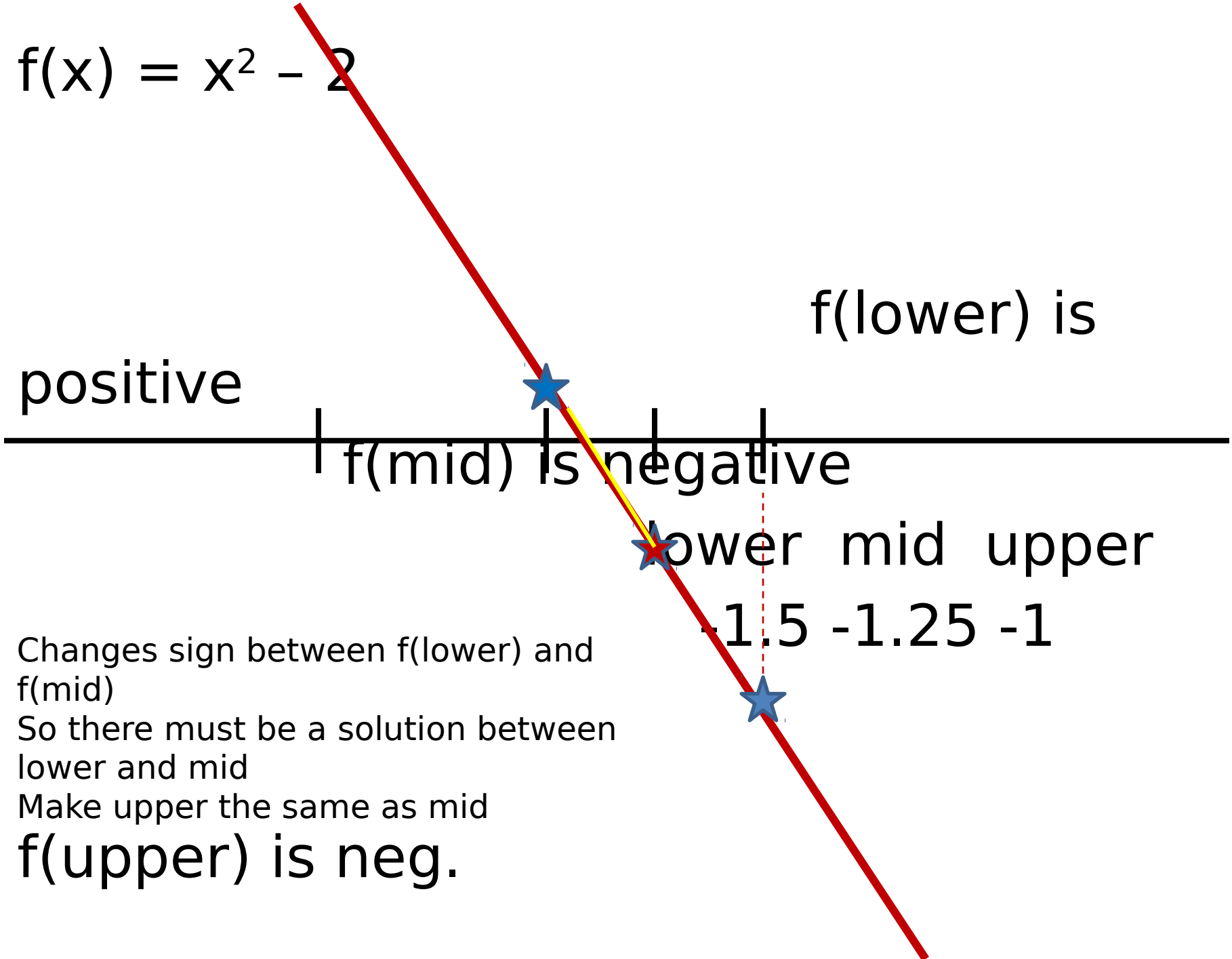

f(x) = x² – 2
f(lower) is positive
lower
-2
upper
-1
Changes sign between f(lower) and f(upper)
So there must be a solution between lower and upper
f(upper) is neg.

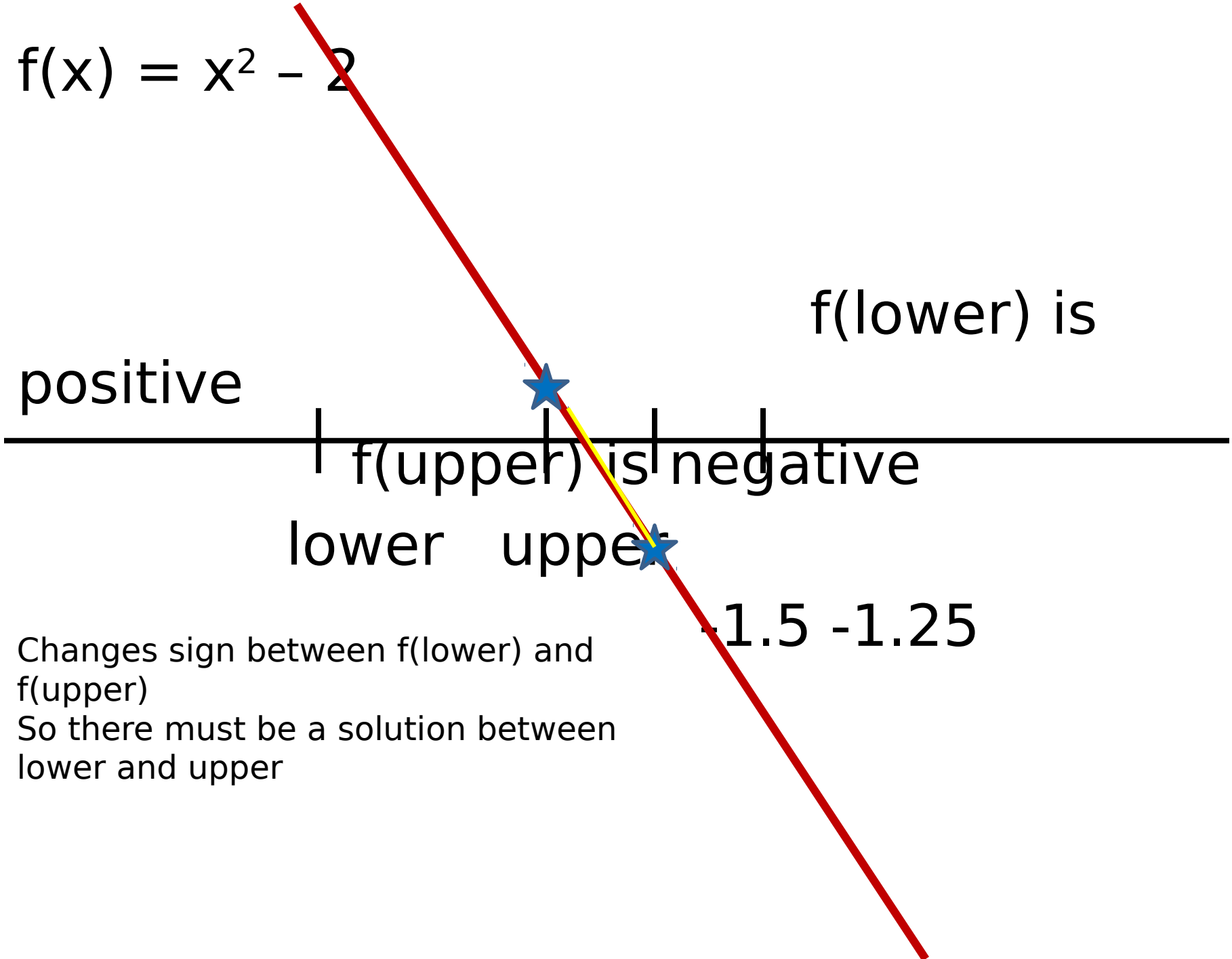


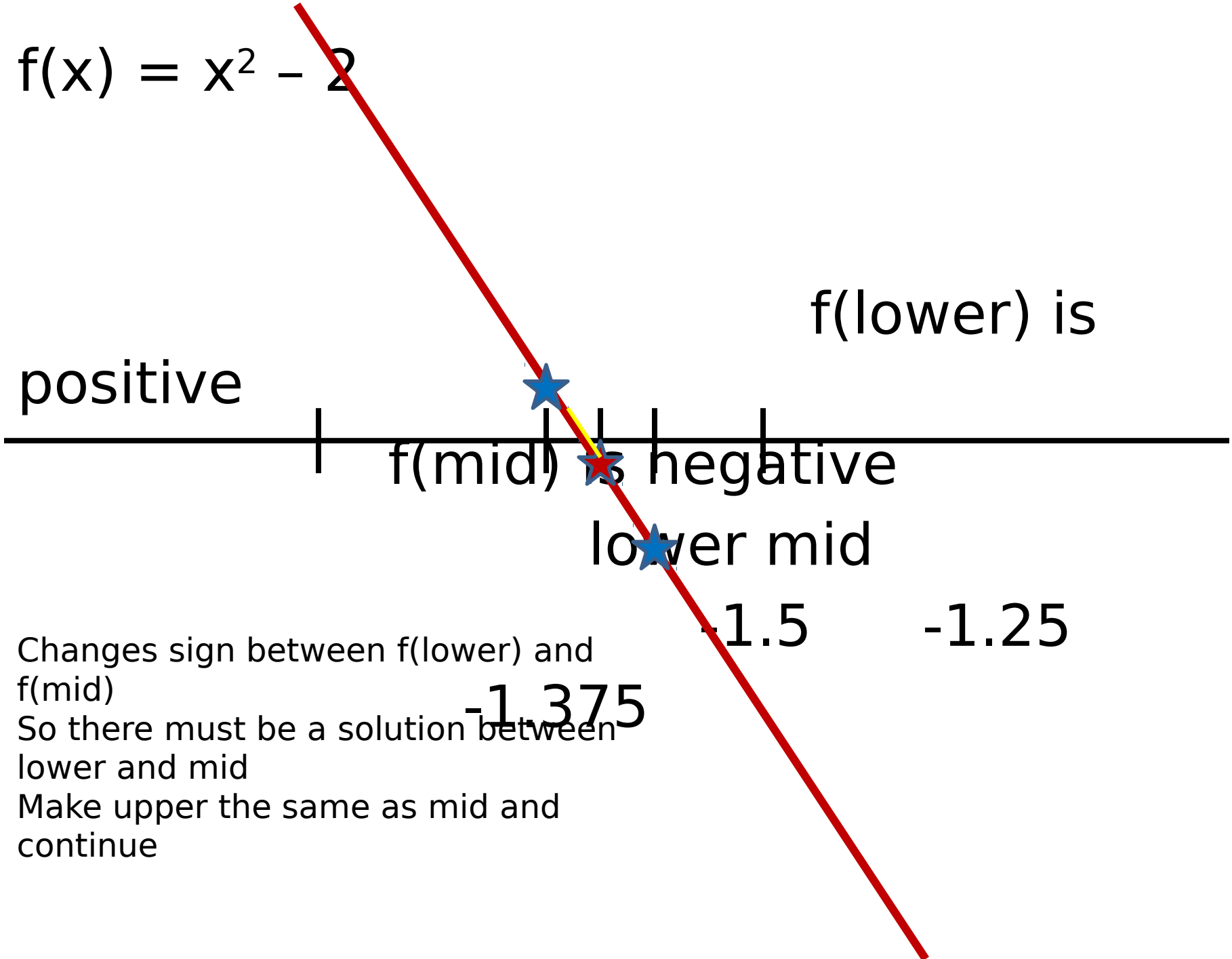
Changes sign between f(mid) and f(upper)
So there must be a solution between mid and upper
Change lower to be the same as mid

f(upper) is neg.


f(x) = x² – 2
f(lower) is
positive
lower
upper
-1.5
-1
Changes sign between f(lower) and f(upper)
So there must be a solution between lower and upper
f(upper) is neg.


f(x) = x² – 2
f(lower) is positive
f(mid) is negative
lower mid upper
-1.5 -1.25 -1
Changes sign between f(lower) and f(mid)
So there must be a solution between lower and mid
Make upper the same as mid
f(upper) is neg.


f(x) = x² – 2
f(lower) is positive
f(upper) is negative
lower
upper
-1.5 -1.25
Changes sign between f(lower) and f(upper)
So there must be a solution between lower and upper



Changes sign between f(lower) and f(mid)
So there must be a solution between lower and mid
Make upper the same as mid and continue