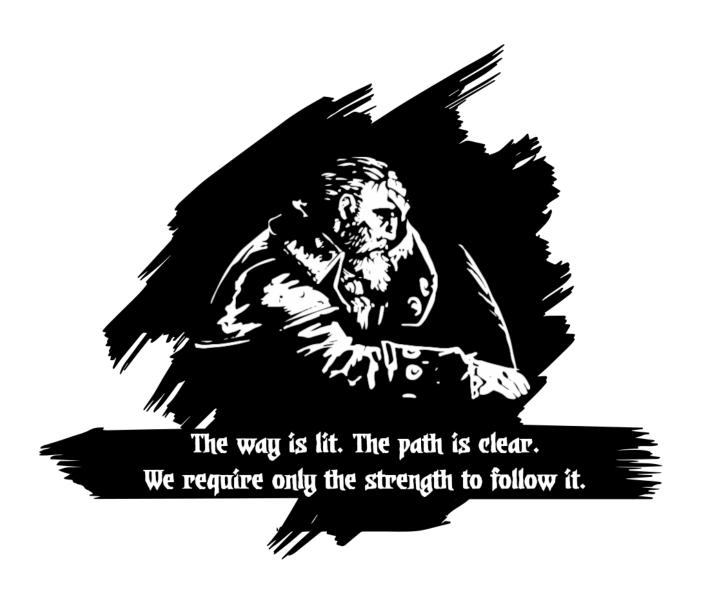


Exercises — Array Binary Search

version #7be580532266ed398481e31366afcc24b1950c2a



Copyright

This document is for internal use at EPITA (website) only.

Copyright © 2022-2023 Assistants <assistants@tickets.assistants.epita.fr>

The use of this document must abide by the following rules:

- ▶ You downloaded it from the assistants' intranet.*
- ▶ This document is strictly personal and must **not** be passed onto someone else.
- ▷ Non-compliance with these rules can lead to severe sanctions.

Contents

1	Goal	3
2	Example	3

^{*}https://intra.assistants.epita.fr

File Tree

```
binary_search/
binary_search.c (to submit)
```

Authorized headers: You are only allowed to use the functions defined in the following headers

err.h

errno.h

assert.h

stddef.h

Compilation: Your code must compile with the following flags

• -std=c99 -pedantic -Werror -Wall -Wextra -Wvla

Main function: None

1 Goal

```
int binary_search(const int vec[], int size, int elt);
```

When looking for an element in a sorted array, it is possible to get the result with a logarithmic complexity using *dichotomy*. Let us remind how dichotomy works:

- If the minimum and maximum indices of the sub-vector are equal or reversed (max <= min), the search is negative (we did not find the element).
- Otherwise, the middle element of the array is chosen as the pivot:
 - If the searched item is equal to the pivot, we return its position.
 - If the searched item is greater than the pivot, we restart the search on the sub-vector starting at the pivot + 1 position.
 - If the searched item is less than the pivot, we restart the search on the sub-vector ending at the pivot position.

2 Example

Write the function that returns the index of the searched element in a sorted vector of integers, or -1 if it is not present. The size of the array will always be correct.

The way is lit. The path is clear. We require only the strength to follow it.