0x01. C - Big O & binary search

System programming & Algorithm — Data structures and Algorithms

by Alexandre Gautier, Junior software engineer at Holberton School

weight: 1

Ongoing project - started 06-12-2017, must end by 06-19-2017 (in 2 days) - you're 0% done.

QA review fully automated.

Readme

Read:

- Search algorithm (https://en.wikipedia.org/wiki/Search_algorithm), Big O notation (http://stackoverflow.com/questions/487258/what-is-a-plain-english-explanation-of-big-o-notation), Space complexity (1) (http://www.geeksforgeeks.org/g-fact-86/) and Space complexity (2) (http://btechsmartclass.com/DS/U1_T3.html).

You can find a lot of documentation about those algorithms on the internet. Don't be afraid to make you own research.

What you should learn from this project

At the end of this project you are expected to be able to explain to anyone, without the help of Google:

- What is the Big O notation, and how to evaluate the time and space complexity of an algorithm
- What is a search algorithm
- What is a linear search
- What is a binary search

Requirements

- Allowed editors: `vi` , `vim` , `emacs`
- All your files will be compiled on Ubuntu 14.04 LTS
- Your programs and functions will be compiled with `gcc 4.8.4` (`C90`) using the flags `-Wall -Werror -Wextra` and `-pedantic`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl (https://github.com/holbertonschool/Betty/blob/master/betty-style.pl) and betty-doc.pl (https://github.com/holbertonschool/Betty/blob/master/betty-doc.pl)
- You are not allowed to use global variables
- No more than 5 functions per file
- You are only allowed to use the `printf` function of the standard library. Any call to another function like `strdup` , `malloc` , ... is forbidden.
- In the following examples, the `main.c` files are showed as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one showed in the examples
- The prototypes of all your functions should be included in your header file called `search_algos.h`
- Don't forget to push your header file
- All your header files should be include guarded

You will be asked to write files containing big O notations. Please use this format:

- `O(1)`
- `O(n)`
- `O(n!)`
- n*m -> `O(nm)`
- n square -> `O(n^2)`
- sqrt n -> `O(sqrt(n))`
- log(n) -> `O(log(n))`
- n * log(n) -> `O(nlog(n))`
- ...

Tasks

0. Big O #0 mandatory

Done?

Help!

What is the time complexity of this function / algorithm?

```
void f(int n)
{
    printf("n = %d\n", n);
}
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-0`

Check your code?

1. Big O #1 mandatory

Done?

Help!

What is the time complexity of this function / algorithm?

```
void f(int n)
{
    int i;

    for (i = 0; i < n; i++)
    {
        printf("[%d]\n", i);
    }
}
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-1`

Check your code?

2. Big O #2 mandatory

Done?

What is the time complexity of this function / algorithm?

Help!

```
void f(int n)
{
    int i;

    for (i = 0; i < n; i += 98)
    {
        printf("[%d]\n", i);
    }
}
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-2`

Check your code?

3. Big O #3 mandatory

Done?

What is the time complexity of this function / algorithm?

Help!

```
void f(unsigned int n)
{
    int i;

    for (i = 1; i < n; i = i * 2)
    {
        printf("[%d]\n", i);
    }
}
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-3`

Check your code?

4. Big O #4 mandatory

Done?

Help!

What is the time complexity of this function / algorithm?

```
var factorial = function(n) {
    if(n == 0) {
        return 1
    } else {
        return n * factorial(n - 1);
    }
}
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-4`

Check your code?

5. Big O #5 mandatory

Done?

Help!

What is the time complexity of this function / algorithm?

```
foreach($numbers as $number)
{
    echo $number;
}
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-5`

Check your code?

6. Big O #6 mandatory

Done?

What is the time complexity of this function / algorithm?

```
void f(unsigned int n)
{
    int i;
    int j;

    for (i = 0; i < n; i++)
    {
        for (j = 1; j < n; j = j * 2)
        {
            printf("[%d] [%d]\n", i, j);
        }
    }
}
```

Help!

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-6`

Check your code?

7. Big O #7 mandatory

What is the time complexity of this function / algorithm?

☐ Done?

Help!

```
void f(int n)
{
    int i;
    int j;

    for (i = 0; i < n; i++)
    {
        if (i % 2 == 0)
        {
            for (j = 1; j < n; j = j * 2)
            {
                printf("[%d] [%d]\n", i, j);
            }
        }
        else
        {
            for (j = 0; j < n; j = j + 2)
            {
                printf("[%d] [%d]\n", i, j);
            }
        }
    }
}
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-7`

Check your code?

8. Big O #8 mandatory

Done?

Help!

What is the time complexity of this function / algorithm?

```
int Fibonacci(int number)
{
    if (number <= 1) return number;

    return Fibonacci(number - 2) + Fibonacci(number - 1);
}
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`

- Directory: `0x01-big-0-binary-search`
- File: `BO-8`

Check your code?

9. Big O #9 mandatory

Done?

Help!

What is the time complexity of this function / algorithm?

```
def func(n):
    a=5
    b=6
    c=10
    for i in range(n):
        for j in range(n):
            x = i * i
            y = j * j
            z = i * j
    for k in range(n):
        w = a*k + 45
        v = b*b
    d = 33
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-9`

Check your code?

10. Big O #10 mandatory

Done?

Help!

What is the time complexity of this function / algorithm?

```
void f(int n)
{
     int i;
     int j;

     for (i = 0; i < n; i++)
     {
          for (j = i + 1; j < n; j++)
          {
               printf("[%d] [%d]\n", i, j);
          }
     }
}
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-10`

Check your code?

11. Big O #Singly linked lists mandatory

Done?

Help!

What are the time complexities of those operations on a singly linked list (one answer per line):

- Accessing the nth element
- Inserting after the nth element (Considering you have a pointer to the node to insert)
- Removing the nth element (Considering you have a pointer to the node to remove)
- Searching for an element in a linked list of size n
- Setting the value of the nth element (Considering you have a pointer to the node to set the value of)

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-11`

Check your code?

12. Big O #Doubly linked lists mandatory

Done?

Help!

What are the time complexities of those operations on a doubly linked list (one answer per line):

- Accessing the nth element
- Inserting after the nth element (Considering you have a pointer to the node to insert)
- Removing the nth element (Considering you have a pointer to the node to remove)
- Searching for an element in a linked list of size n
- Setting the value of the nth element (Considering you have a pointer to the node to set the value of)

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-12`

Check your code?

13. Big O #Hash tables mandatory

Done?

Help!

What are the time complexities of those operations on a hash table (one answer per line) - with the implementation you used during the previous Hash Table C project (chaining):

- Searching for an element, best case
- Searching for an element, worst case
- Insertion, best case
- Insertion, worst case
- Deletion, best case
- Deletion, worst case

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-15`

Check your code?

14. Big O #16 mandatory

Done?

Help!

What is the space complexity of this function / algorithm?

```
int **allocate_map(int n, int m)
{
     int **map;

     map = malloc(sizeof(int *) * n);
     for (size_t i = 0; i < n; i++)
     {
          map[i] = malloc(sizeof(int) * m);
     }
     return (map);
}
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-16`

Check your code?

15. Linear search mandatory

Done?

Help!

Write a function that searches for a value in an array of integers using the Linear search algorithm (https://en.wikipedia.org/wiki/Linear_search)

- Prototype : `int linear_search(int *array, size_t size, int value);`
- Where `array` is a pointer to the first element of the array to search in
- `size` is the number of elements in `array`
- And `value` is the value to search for
- Your function must return the first index where `value` is located
- If `value` is not present in `array` or if `array` is `NULL` , your function must return `-1`
- Every time you compare a value in the array to the value you are searching, you have to print this value (see example below)

```
wilfried@0x1D-search_algorithms$ cat 0-main.c
#include <stdio.h>
#include <stdlib.h>
#include "search_algos.h"

/**
 * main - Entry point
 *
 * Return: Always EXIT_SUCCESS
 */
int main(void)
{
    int array[] = {
        10, 1, 42, 3, 4, 42, 6, 7, -1, 9
    };
    size_t size = sizeof(array) / sizeof(array[0]);

    printf("Found %d at index: %d\n\n", 3, linear_search(array, size, 3));
    printf("Found %d at index: %d\n\n", 42, linear_search(array, size, 42));
    printf("Found %d at index: %d\n", 999, linear_search(array, size, 999));
    return (EXIT_SUCCESS);
}
wilfried@0x1D-search_algorithms$ gcc -Wall -Wextra -Werror -pedantic 0-main.c 0-li
near.c -o 0-linear
wilfried@0x1D-search_algorithms$ ./0-linear
Value checked array[0] = [10]
Value checked array[1] = [1]
Value checked array[2] = [42]
Value checked array[3] = [3]
Found 3 at index: 3

Value checked array[0] = [10]
Value checked array[1] = [1]
Value checked array[2] = [42]
Found 42 at index: 2

Value checked array[0] = [10]
Value checked array[1] = [1]
Value checked array[2] = [42]
Value checked array[3] = [3]
Value checked array[4] = [4]
Value checked array[5] = [42]
Value checked array[6] = [6]
Value checked array[7] = [7]
Value checked array[8] = [-1]
Value checked array[9] = [9]
Found 999 at index: -1
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`

- Directory: `0x01-big-0-binary-search`
- File: `0-linear.c`

Check your code?

16. Binary search mandatory

Done?

Help!

Write a function that searches for a value in a sorted array of integers using the Binary search algorithm (https://en.wikipedia.org/wiki/Binary_search_algorithm)

- Prototype : `int binary_search(int *array, size_t size, int value);`
- Where `array` is a pointer to the first element of the array to search in
- `size` is the number of elements in `array`
- And `value` is the value to search for
- Your function must return the index where `value` is located
- You can assume that `array` will be sorted in ascending order
- You can assume that `value` won't appear more than once in `array`
- If `value` is not present in `array` or if `array` is `NULL` , your function must return `-1`
- Every time you split the array, you have to print the new array (or subarray) you're searching in (See example)

```
wilfried@0x1D-search_algorithms$ cat 1-main.c
#include <stdio.h>
#include <stdlib.h>
#include "search_algos.h"

/**
 * main - Entry point
 *
 * Return: Always EXIT_SUCCESS
 */
int main(void)
{
    int array[] = {
        0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    };
    size_t size = sizeof(array) / sizeof(array[0]);

    printf("Found %d at index: %d\n\n", 2, binary_search(array, size, 2));
    printf("Found %d at index: %d\n\n", 5, binary_search(array, 4, 5));
    printf("Found %d at index: %d\n", 999, binary_search(array, size, 999));
    return (EXIT_SUCCESS);
}
wilfried@0x1D-search_algorithms$ gcc -Wall -Wextra -Werror -pedantic 1-main.c 1-bi
nary.c -o 1-binary
wilfried@0x1D-search_algorithms$ ./1-binary
Searching in array: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Searching in array: 0, 1, 2, 3, 4
Found 2 at index: 2

Searching in array: 0, 1, 2, 3
Searching in array: 2, 3
Searching in array: 3
Found 5 at index: -1

Searching in array: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Searching in array: 5, 6, 7, 8, 9
Searching in array: 8, 9
Searching in array: 9
Found 999 at index: -1
```

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `1-binary.c`

Check your code?

17. Big O #Linear search - time complexity mandatory

Done?

Help!

What is the time complexity of a basic linear search algorithm in an array of size n?

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-100`

Check your code?

18. Big O #Linear search - space complexity mandatory

Done?

Help!

What is the space complexity of a basic linear search algorithm in an array of size n?

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-101`

Check your code?

19. Big O #Binary search - time complexity mandatory

Done?

Help!

What is the time complexity of a basic binary search algorithm of an array of size n?

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-102`

Check your code?

20. Big O #Binary search - space complexity mandatory

Done?

Help!

What is the space complexity of a basic binary search algorithm of an array of size n?

Repo:

- GitHub repository: `holbertonschool-datastructures_algorithms`
- Directory: `0x01-big-0-binary-search`
- File: `BO-103`

Check your code?

Done with the mandatory tasks? Unlock 2 advanced tasks now! (/projects/318/unlock_optionals)