EPITECH PARIS
2023

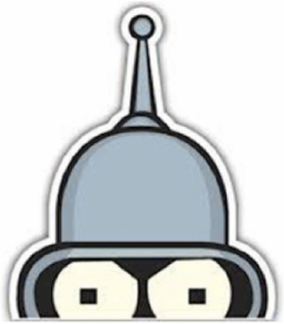
Review
B-CPE-110: BSQ

TEAM: VERA KOLIVERDA

EPITECH PARIS
2023

BSQ

find the Biggest SQuare



Formal part of the project description

Project unit: B-CPE-110
Project name: BSQ
Duration: 3 weeks
Project type: individual project

EPITECH PARIS
2023

BSQ Project description

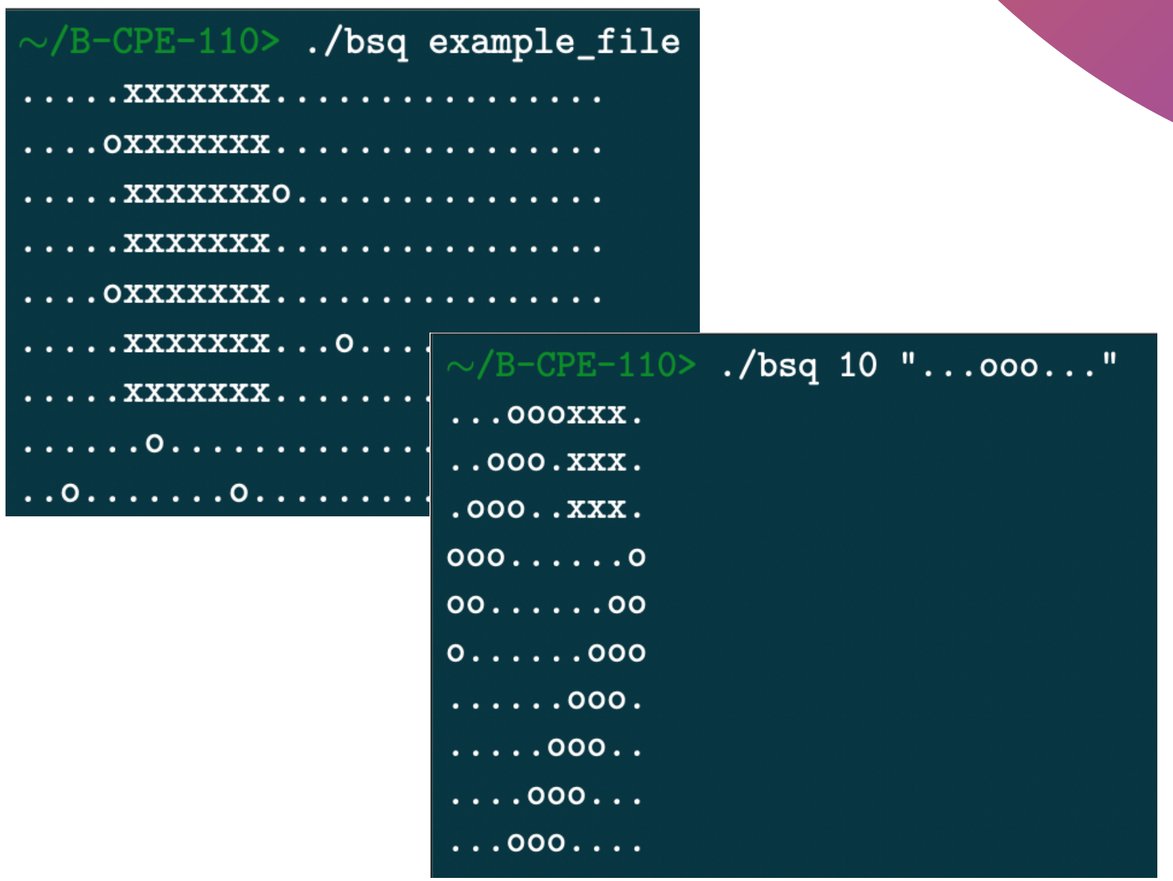
Given a "map" as input, we return the same map with the biggest square marked with stars.

What is a map?

- it is a line containing '.', 'o' and new line characters only
- 'o' stands for obstacles

Two ways of getting a map:

- generate map based on a pattern
- read map from file



EPITECH PARIS
2023

Notions and programming patterns used

01 SOLID principles

- Project code is easy-to-read
- The code is easy to upscale
- The code is simple but efficient
- Easy to debug

02 Broot force approach

- Trying to achieve the task by crossing all the map and scanning for a square at each non obstacle character
- Easiest to understand and implement
- Not efficient

03 Dynamic programming approach

- More efficient approach that I had to research and implement
- Not easy to get used to it and implement from scratch

EPITECH PARIS
2023

Project structure

- To build an easy to read project structure, I used a typical C project template with include, src, test directories and Makefile
- Two blocks inside src:
  - map_structure: contains all tools to initialise and scan map
  - squares: contains all tools to scan initialised map and draw resulting square
- Library:
  - contains classic lib self-made functions and some tools to operate the map structure

```
├── Makefile
├── coding-style-reports.log
├── include
│   └── my.h
├── lib
│   ├── display_map.c
│   ├── free_map.c
│   ├── generate_content.c
│   ├── get_rows_cols.c
│   ├── mem_alloc_2d_array.c
│   ├── my_putchar.c
│   ├── my_putstr.c
│   ├── read_file_to_string.c
│   └── string_to_int.c
├── src
│   ├── bsq.c
│   ├── main.c
│   ├── map_structure
│   │   ├── browse_map.c
│   │   ├── map_from_file_init.c
│   │   ├── map_generate_init.c
│   │   └── parse_content.c
│   └── squares
│       ├── draw_square.c
│       └── find_biggest_square.c
```

```c
typedef struct my_map {
    char const *filepath;
    char const *pattern;
    char *content;
    int nb_rows;
    int nb_cols;
    int max_row;
    int max_col;
    int max_square_size;
    char **init;
    int **result;
} my_map;
```

EPITECH PARIS
2023

Steps of the project and its timeline

01

Solving Bootstrap of BSQ

- Allowed me to understand the task better
- I obtained ready-to-implement functions for the main part of the project
- It took **2 or 3 days**

02

Version 1 of the project

- Implementing broot force approach
- Passing almost 85 percent of the tests
- Only those for robustness do not pass for a big map
- I took **5 days**

02

Version 2 of the project

- Learning algorithms and data structures on my own, I decided to try this approach to make a more efficient algorithm
- Passing almost 95 percents of the tests
- It took about **5 days**

EPITECH PARIS
2023

Final algorithm: dynamic programming

```
                c1   c2
_____________________
|                |     |   |
|   A            | B   |   |
|________________|_____|   |   r1
|                |     |   |
|     C          |   D |   |
|________________|_____|   |   r2
|_____________________|
```

We want the count of stuff inside D. In terms of our pre-computed counts from the top left.

$$Count(D) = Count(A \cup B \cup C \cup D) - Count(A \cup C) - Count(A \cup B) + Count(A)$$

Source: https://stackoverflow.com/questions/20335427/most-efficient-algorithm-to-find-the-biggest-square-in-a-two-dimension-map

EPITECH PARIS
2023

Problems & solutions

Initialising a map

- It was not evident how to choose the best structure for the map
- Solved by testing different approaches

Memory allocation

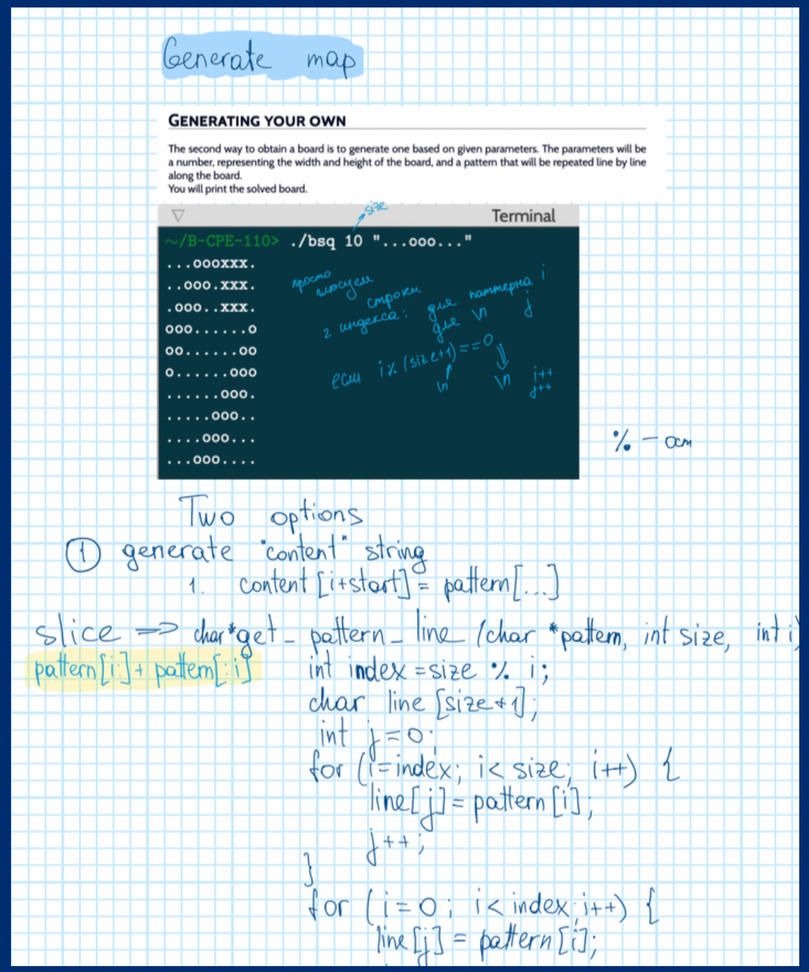
- had some memory errors related to poor memory management
- solved with debugging using valgrind at each project iteration

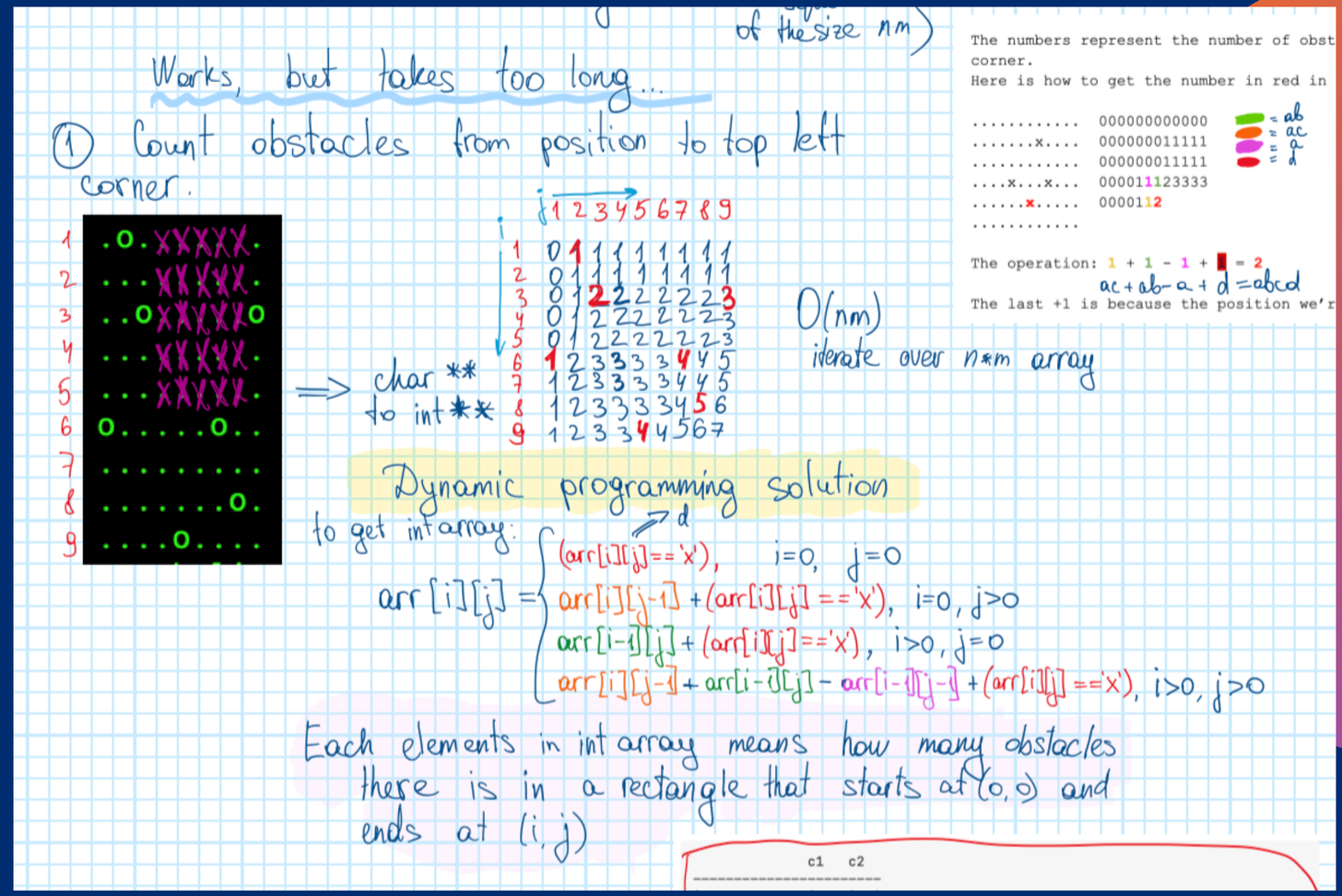
Increase efficiency

- dynamic programming approach was difficult to understand and implement
- solved by visualising everything that is not really clear

EPITECH PARIS
2023

Visualising to understand better





EPITECH PARIS
2023

Final results

BSQ

01/01/2023 05:45

Prerequisites met

Coding style

Major	0
Minor	0
Info	0

Coverage

Data structure - Generating

100% Passed

Total: 3
Passed: 3
Crashed: 0
Failed or skipped: 0

Data structure - Opening

100% Passed

Total: 3
Passed: 3
Crashed: 0
Failed or skipped: 0

Optimization - Generating 100 to 500

100% Passed

Total: 4
Passed: 4
Crashed: 0
Failed or skipped: 0

Optimization - Generating 1000 to 2000

100% Passed

Total: 2
Passed: 2
Crashed: 0
Failed or skipped: 0

Optimization - Generating 5000 to 10000

0% Passed

Total: 2
Passed: 0
Crashed: 0
Failed or skipped: 2

EPITECH PARIS
2023

Live demonstration & conclusion

I have made a research of a dynamic approach solution that helped me achieve a high score

STUDIO SHODWE

Thank You