Compression and Decompression Tool

Gaumart Siméon
0845209

Introduction

- compression and decompression is an important problematic :

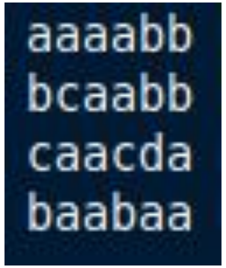
  -> Optimize storage

  -> Send more data (increase throughput)

- Many compression algorithm : RLE, Huffman code, LZ77, (deflate) ...

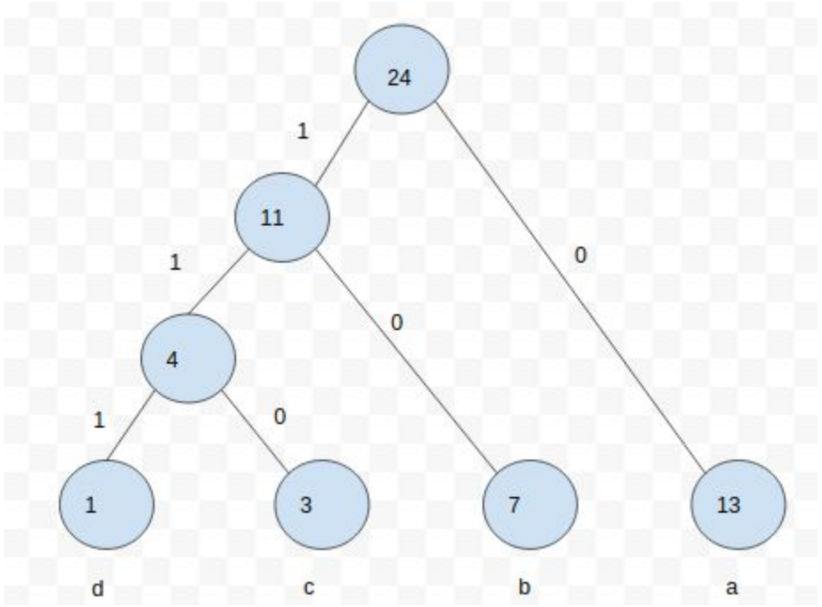
Introduction

Huffman Code



aaaabb
bcaabb
caacda
baabaa

13 x a
7 x b
3 x c
1 x d



Huffman Code

```
class HuffmanTree{

        std::vector<struct node> tree;

        struct node * root;

        *constructor*(std::string text);

        std::vector<struct codage> get_code();
};
```

```
struct node{
        char leaf;      //NULL for no leaf
        signed long long freq;
        struct arc right; //NULL for leaf
        struct arc left; //NULL for leaf
        struct arc * root; //NULL for root
        size_t pass=0;
};
```

```
struct arc{
    char bin;
    struct node * father;
    struct node * child;
};
```

Huffman Code

```
*constructor*(text){
        leafs=get_frequency(text)
        * roots_list=[leafs]
        tree.push_back(leafs)

        while roots_list.size() != 1 {
                min1,min2=get_mins(roots_list)
                new_node
                new_node.right=min1;new_node.left=min2
                new_node.freq=min1.freq+min2.freq
                tree.push_back(new_node)
                roots_list.push_back(*new_node)
                roots_list.remove(min1,min2)
        }
        root=roots_list[0]
}
```

```
std:vector<struct codage> get_node(){
        perform DFS
        (Depth-first search)

                or variant

        get bin for each arc
}
```

```
struct codage{
   char leaf;
   std::string bin;
};
```

CDT

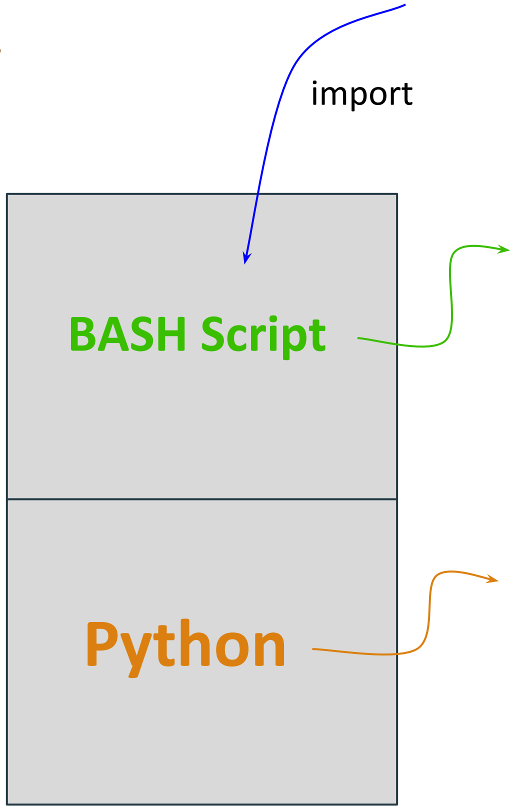
command line : **./cdt**

options (needed) :

-d	for using decompression	**-m**	for choosing module (cpp file or module name)
-c	for using compression	**-f**	for choosing the file(s) (1 file or multiple file with “file1 file2 ...”)

CDT

Module (for compression and decompression)

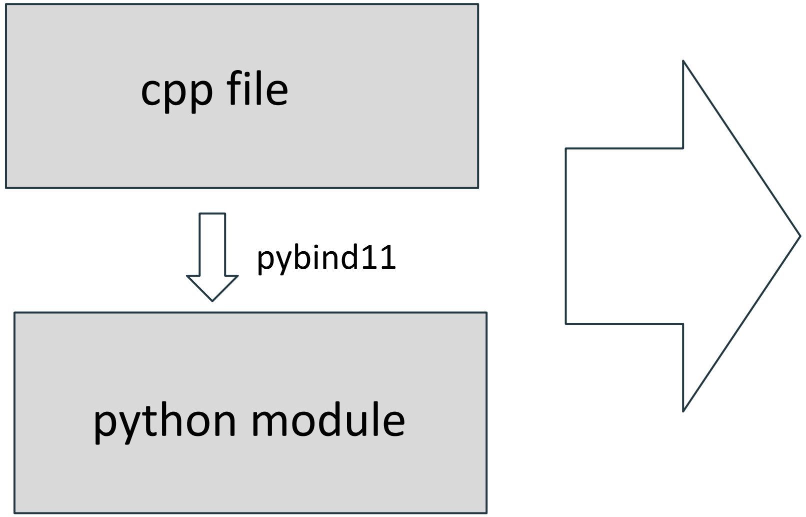


-> recover all arguments (getopts)
-> build **module** from cpp file (if cpp file)
-> check if the **module** can be imported
-> check if it's compression or decompression
-> import **module** for python
-> run python code (using python3 or python2)

-> iterate through all files for compression (or decompression)
-> check if the file exist
-> check if the file created by compression (or decompression) already exist
-> read/write binary into files
-> call compression (or decompression) from **module**

CDT

Module (for compression and decompression) :



hexadecimal code with a string

content of text file

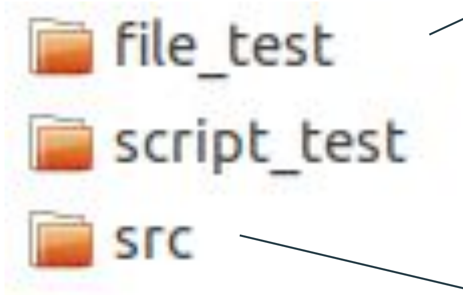
```
string compression(string text);

string decompression(string hexa);
```

reverse of compression

CDT

Tests :



file_test

script_test

src

. txt files (some files with 1 character, 1 file with a text from a book)

. python tests (runnable with pytest)

. Makefile (test, clean)
. module file (.cpp or .so)
    -> testcompression
    -> Hufmman
. (cdt)

Conclusion

Perform Compression and Decompression over text files

Use any Compression and Decompression algorithm since it’s suitable with cdt

Perform test easily with pytest

Implemented Huffman algorithm for compression and decompression (almost)

Conclusion

What to improve ?

. make Huffman runnable

. make cdt runnable for every file type (and directory)

. change input/output for compression/decompression

. manage non-factor of 8 size for binary result (compression)

. install cdt

. all documentation

And many things….

Thank you for your attention