Lab 1 Sorting

1.0

Generated by Doxygen 1.9.5

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Generator Class Reference

Public Member Functions

- Model **model_generator** ()

The documentation for this class was generated from the following files:

- src/generator/generator.hpp
- src/generator/generator.cpp

3.2 Model Class Reference

Public Member Functions

- **Model** (std::string full_name, std::string department, std::string job_title, std::chrono::year_month_day employment_date)
- **Model** (std::string full_name, std::string department, std::string job_title, std::string employment_date)
- **Model** (std::uint8_t decor_type)
- void **set_model** (std::string full_name, std::string department, std::string job_title, std::chrono::year_month↩ _day employment_date)
- void **set_model** (std::string full_name, std::string department, std::string job_title, std::string employment_↩ date)
- void **set_decor** (std::uint8_t decor_type)
- ModelComp **compare_type** (const Model &r_model, uint8_t mode)

Static Public Member Functions

- static void **save_model** (const std::vector< Model > &model_vector, std::filesystem::path file_path)
- static void **load_model** (std::vector< Model > &model_vector, std::filesystem::path file_path)
- static void **print_model** (const std::vector< Model > &model_vector)

Friends

- std::ostream & **operator<<** (std::ostream &stream, const Model &model)
- ModelComp **operator<** (const Model &l_model, const Model &r_model)
- ModelComp **operator>** (const Model &l_model, const Model &r_model)
- ModelComp **operator<=** (const Model &l_model, const Model &r_model)
- ModelComp **operator>=** (const Model &l_model, const Model &r_model)
- ModelComp **operator==** (const Model &l_model, const Model &r_model)
- ModelComp **operator!=** (const Model &l_model, const Model &r_model)

The documentation for this class was generated from the following files:

- src/model/model.hpp
- src/model/model.cpp

3.3 ModelComp Class Reference

Public Member Functions

- **ModelComp** (uint8_t value)
- void **set_type_masked** (uint8_t value, uint8_t offset)
- void **set_name_type** (uint8_t value)
- void **set_dept_type** (uint8_t value)
- void **set_jobt_type** (uint8_t value)
- void **set_date_type** (uint8_t value)
- uint8_t **get_type_masked** (uint8_t offset) const
- uint8_t **get_name_type** () const
- uint8_t **get_dept_type** () const
- uint8_t **get_jobt_type** () const
- uint8_t **get_date_type** () const
- ModelComp **operator!** () const

Friends

- ModelComp **operator==** (const ModelComp &l_bool, const ModelComp &r_bool)
- ModelComp **operator!=** (const ModelComp &l_bool, const ModelComp &r_bool)
- std::ostream & **operator<<** (std::ostream &stream, const ModelComp &model)

The documentation for this class was generated from the following files:

- src/model/model.hpp
- src/model/model.cpp

3.4 Sorting Class Reference

A class that provides static sorting methods for sorting a vector of Model objects based on a specific field.

```
#include <sorting.hpp>
```

Static Public Member Functions

- static void bubble_sort (std::vector< Model > &model_vector, uint8_t field)

  Sorts the given vector of Model objects using bubble sort algorithm.
- static void heap_sort (std::vector< Model > &model_vector, uint8_t field)

  Performs heap sort on a vector of Model objects.
- static void merge_sort (std::vector< Model > &model_vector, uint8_t field, std::size_t left=0, std::size_↩
t right=0, bool initial=true)

  Sorts a vector of Model objects using merge sort algorithm.

3.4.1 Detailed Description

A class that provides static sorting methods for sorting a vector of Model objects based on a specific field.

Note

Currently provides implementations for bubble sort, heap sort, and merge sort.

3.4.2 Member Function Documentation

3.4.2.1 bubble_sort()

```
void Sorting::bubble_sort (
             std::vector< Model > & model_vector,
             uint8_t field )  [static]
```

Sorts the given vector of Model objects using bubble sort algorithm.

This function uses bubble sort algorithm to sort the given vector of Model objects based on the field specified by the `field` parameter. The objects are compared using the `compare_type()` method of the Model class.

Parameters

model_vector	The vector of Model objects to be sorted.
field	The index of the field to be used for sorting the objects.

Returns

void.

Note

This function modifies the original vector passed to it.

The `compare_type()` method of the Model class must return a Type object.

The Type object must have a method `get_type_masked()` that takes an offset and returns a boolean indicating whether the specified bit is set or not.

This function prints the number of iterations taken to sort the vector.

3.4.2.2 heap_sort()

```
void Sorting::heap_sort (
            std::vector< Model > & model_vector,
            uint8_t field )  [static]
```

Performs heap sort on a vector of Model objects.

This function sorts a vector of Model objects using the heap sort algorithm. The function uses the make_heap function to create a heap from the input vector, then sorts the heap by repeatedly extracting the maximum element from the heap and placing it at the end of the vector.

Parameters

model_vector	The vector of Model objects to be sorted.
field	The field of the Model object to sort by.

Returns

void.

3.4.2.3 merge_sort()

```
void Sorting::merge_sort (
            std::vector< Model > & model_vector,
            uint8_t field,
            std::size_t left = 0,
            std::size_t right = 0,
            bool initial = true )  [static]
```

Sorts a vector of Model objects using merge sort algorithm.

This function sorts a given vector of Model objects using merge sort algorithm. It takes the field to be sorted as input, along with left and right indices of the sub-vector to be sorted. If left and right indices are not provided, it sorts the entire vector by setting left and right indices accordingly.

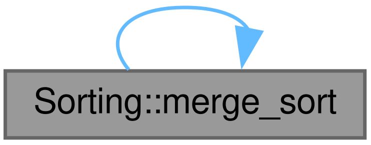
Parameters

model_vector	The vector of Model objects to be sorted.
field	The field to be sorted.
left	The left index of the sub-vector to be sorted (default is 0).
right	The right index of the sub-vector to be sorted (default is size-1).
initial	A boolean flag indicating whether this is the initial call to the function (default is true).

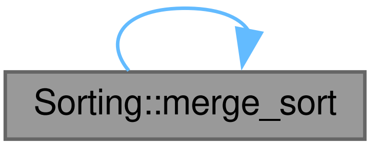
Returns

void.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- src/sorting/sorting.hpp
- src/sorting/sorting.cpp

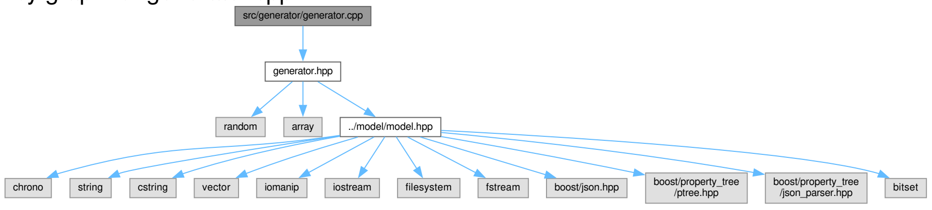
Chapter 4

File Documentation

4.1 src/generator/generator.cpp File Reference

This source file holds implementation of Generator class.

```
#include "generator.hpp"
```
Include dependency graph for generator.cpp:



4.1.1 Detailed Description

This source file holds implementation of Generator class.

>

```
This calss implements model generator needed for successfull completion of laboratory work 1.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

Copyright

Copyright 2023 Alexander. All rights reserved.

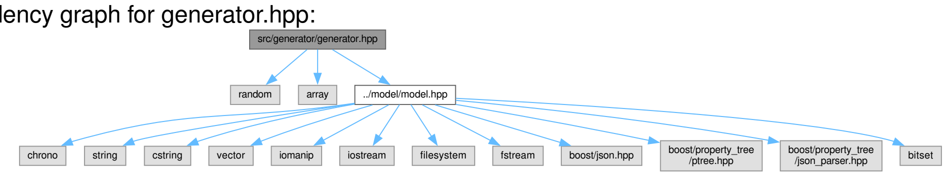
```
(Not really)
```

4.2 src/generator/generator.hpp File Reference

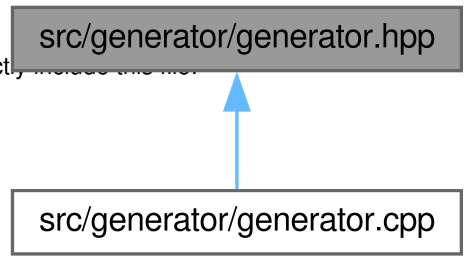
This header file holds implementation of Generator class.

```
#include <random>
#include <array>
#include "../model/model.hpp"
```

Include dependency graph for generator.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Generator

4.2.1 Detailed Description

This header file holds implementation of Generator class.

>

```
This calss implements model generator needed for successfull completion of laboratory work 1.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

Copyright

Copyright 2023 Alexander. All rights reserved.

```
(Not really)
```

4.3 generator.hpp

Go to the documentation of this file.

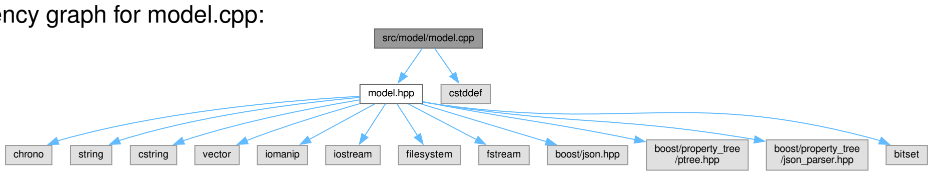
```
#ifndef GENERATOR_HPP
#define GENERATOR_HPP

#include <random>

#include <array>


#ifndef MODEL_HPP
#include "../model/model.hpp"
#endif // MODEL_HPP

class Generator
{
public:
    Generator();
    ~Generator();

    Model model_generator();

private:
    std::array<std::string, 2738> _first_name_list;
    std::array<std::string, 1000> _last_name_list;
    std::array<std::string, 449> _department_list;
    std::array<std::string, 357> _job_title_list;

    std::random_device       _random_device;
    std::mt19937             _generator;

    std::uniform_int_distribution<uint32_t> _first_name_distribution;
    std::uniform_int_distribution<uint32_t> _last_name_distribution;
    std::uniform_int_distribution<uint32_t> _department_distribution;
    std::uniform_int_distribution<uint32_t> _job_title_distribution;
    std::uniform_int_distribution<uint16_t> _year_distribution;
    std::uniform_int_distribution<uint16_t> _month_distribution;
    std::uniform_int_distribution<uint16_t> _day_distribution;
    std::uniform_int_distribution<uint16_t> _sex_distribution;
};

#endif // GENERATOR_HPP
```

4.4 src/model/model.cpp File Reference

This source file holds implementation of Model class.

```
#include "model.hpp"
#include <cstddef>
```

Include dependency graph for model.cpp:



Functions

- std::ostream & **operator<<** (std::ostream &stream, const Model &model)
- ModelComp **operator<** (const Model &l_model, const Model &r_model)
- ModelComp **operator>** (const Model &l_model, const Model &r_model)
- ModelComp **operator<=** (const Model &l_model, const Model &r_model)
- ModelComp **operator>=** (const Model &l_model, const Model &r_model)
- ModelComp **operator==** (const Model &l_model, const Model &r_model)
- ModelComp **operator!=** (const Model &l_model, const Model &r_model)
- ModelComp **operator==** (const ModelComp &l_bool, const ModelComp &r_bool)
- ModelComp **operator!=** (const ModelComp &l_bool, const ModelComp &r_bool)
- std::ostream & **operator<<** (std::ostream &stream, const ModelComp &r_bool)

4.4.1 Detailed Description

This source file holds implementation of Model class.

>

```
    This calss implements model needed for successfull completion of laboratory work 1.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
    In order to submit new ones, please contact me via admin@redline-software.xyz.
```

Copyright

Copyright 2023 Alexander. All rights reserved.

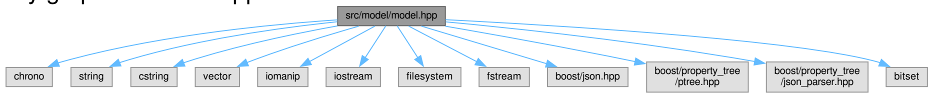
```
        (Not really)
```

4.5 src/model/model.hpp File Reference

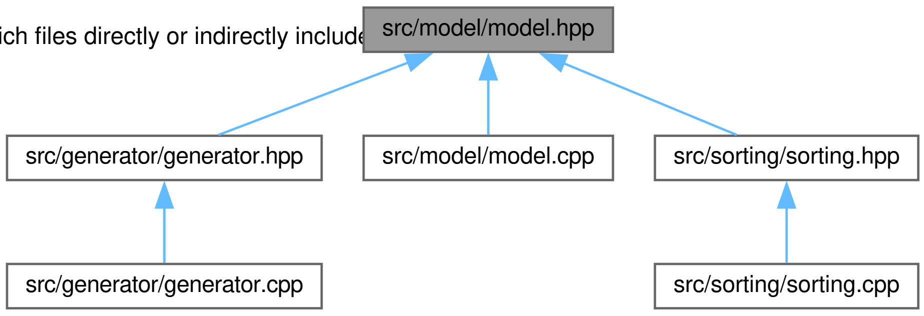
This header file holds implementation of Model class.

```
#include <chrono>
#include <string>
#include <cstring>
#include <vector>
#include <iomanip>
#include <iostream>
#include <filesystem>
#include <fstream>
#include <boost/json.hpp>
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#include <bitset>
```

Include dependency graph for model.hpp:



This graph shows which files directly or indirectly include



Classes

- class Model
- class ModelComp

4.5.1 Detailed Description

This header file holds implementation of Model class.

>

```
This calss implements model needed for successfull completion of laboratory work 1.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

Copyright

Copyright 2023 Alexander. All rights reserved.

```
(Not really)
```

4.6 model.hpp

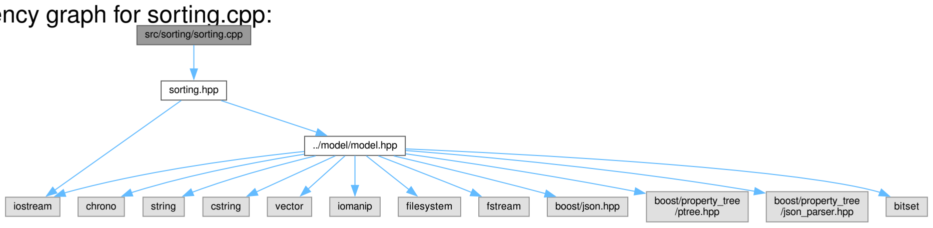
Go to the documentation of this file.

```cpp

#ifndef MODEL_HPP
#define MODEL_HPP

#include <chrono>
#include <string>
#include <cstring> // strcmp has better performance
#include <vector>
#include <iomanip>
#include <iostream>


#include <filesystem>
#include <fstream>
#include <boost/json.hpp>
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>

#include <bitset>

class ModelComp;

class Model
{
public:
    Model(std::string full_name, std::string department, std::string job_title,
    std::chrono::year_month_day employment_date);
    Model(std::string full_name, std::string department, std::string job_title, std::string
    employment_date);
    Model(std::uint8_t decor_type);
    ~Model();

    void set_model(std::string full_name, std::string department, std::string job_title,
    std::chrono::year_month_day employment_date);
    void set_model(std::string full_name, std::string department, std::string job_title, std::string
    employment_date);
    void set_decor(std::uint8_t decor_type);

    ModelComp compare_type(const Model& r_model, uint8_t mode);

    static void save_model(const std::vector<Model>& model_vector, std::filesystem::path file_path);
    static void load_model(std::vector<Model>& model_vector, std::filesystem::path file_path);
    static void print_model(const std::vector<Model>& model_vector);

    friend std::ostream& operator« (std::ostream& stream, const Model& model);

    friend ModelComp     operator<  (const Model& l_model, const Model& r_model);
    friend ModelComp     operator>  (const Model& l_model, const Model& r_model);
    friend ModelComp     operator<= (const Model& l_model, const Model& r_model);
    friend ModelComp     operator>= (const Model& l_model, const Model& r_model);

    friend ModelComp     operator== (const Model& l_model, const Model& r_model);
    friend ModelComp     operator!= (const Model& l_model, const Model& r_model);

private:
    std::string                     _full_name;
    std::string                     _department;
    std::string                     _job_title;
    std::chrono::year_month_day     _employment_date;

    std::uint8_t                    _decor_type;
};

class ModelComp
{
public:
    ModelComp();
    ModelComp(uint8_t value);
    ~ModelComp();

    void set_type_masked(uint8_t value, uint8_t offset);

    void set_name_type(uint8_t value);
    void set_dept_type(uint8_t value);
    void set_jobt_type(uint8_t value);
    void set_date_type(uint8_t value);

    uint8_t get_type_masked(uint8_t offset) const;

    uint8_t get_name_type() const;
    uint8_t get_dept_type() const;
    uint8_t get_jobt_type() const;
    uint8_t get_date_type() const;

    friend ModelComp operator== (const ModelComp& l_bool, const ModelComp& r_bool);
     friend ModelComp operator!= (const ModelComp& l_bool, const ModelComp& r_bool);

     friend std::ostream& operator« (std::ostream& stream, const ModelComp& model);

     ModelComp operator! () const;

private:
     uint8_t _value;
};

#endif // MODEL_HPP
```

4.7 src/sorting/sorting.cpp File Reference

This source file holds implementation of Sorting class.

`#include "sorting.hpp"`
Include dependency graph for sorting.cpp:



4.7.1 Detailed Description

This source file holds implementation of Sorting class.

>

```
This calss implements sorting algorithms needed for successfull completion of laboratory work 1.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

Copyright

Copyright 2023 Alexander. All rights reserved.

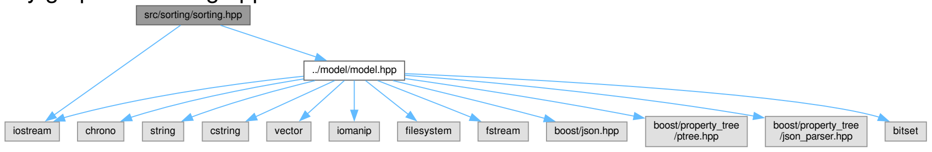
```
(Not really)
```

4.8 src/sorting/sorting.hpp File Reference

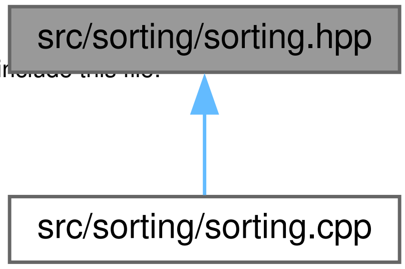
This header file holds implementation of Sorting class.

```
#include "../model/model.hpp"
#include <iostream>
```

Include dependency graph for sorting.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Sorting

    A class that provides static sorting methods for sorting a vector of Model objects based on a specific field.

4.8.1 Detailed Description

This header file holds implementation of Sorting class.

>

```
    This calss implements sorting algorithms needed for successfull completion of laboratory work 1.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

Copyright

Copyright 2023 Alexander. All rights reserved.

```
(Not really)
```

4.9 sorting.hpp

Go to the documentation of this file.

```
#ifndef SORTING_HPP
#define SORTING_HPP

#ifndef MODEL_HPP
#include "../model/model.hpp"
#endif // MODEL_HPP

#include <iostream>

class Sorting
{
public:
    static void bubble_sort(std::vector<Model>& model_vector, uint8_t field);
    static void heap_sort(std::vector<Model>& model_vector, uint8_t field);
    static void merge_sort(std::vector<Model>& model_vector, uint8_t field, std::size_t left = 0,
    std::size_t right = 0, bool initial = true);

private:
    static void make_heap(std::vector<Model>& model_vector, std::size_t index, uint8_t field, std::size_t
    last_index = 0);
    static void make_merge(std::vector<Model>& model_vector, uint8_t field, std::size_t left, std::size_t
    right, std::size_t middle);
};

#endif // SORTING_HPP
```

Chapter 5

Sorting statistics

5.0.1 Definition for sorting

FULL NAME	DEPARTMENT	JOB TITLE	DATE
MODE 0	**MODE 1**	**MODE 2**	**MODE 3**
Alexzander Oliver Baxterovna	Data Entry	Architectural Technologist	2015/09/09
...	...	...	...
Billy Barrett Okeogheneovich	Audio Engineering	Production Manager	1999/01/15

Table 5.1 Employee information

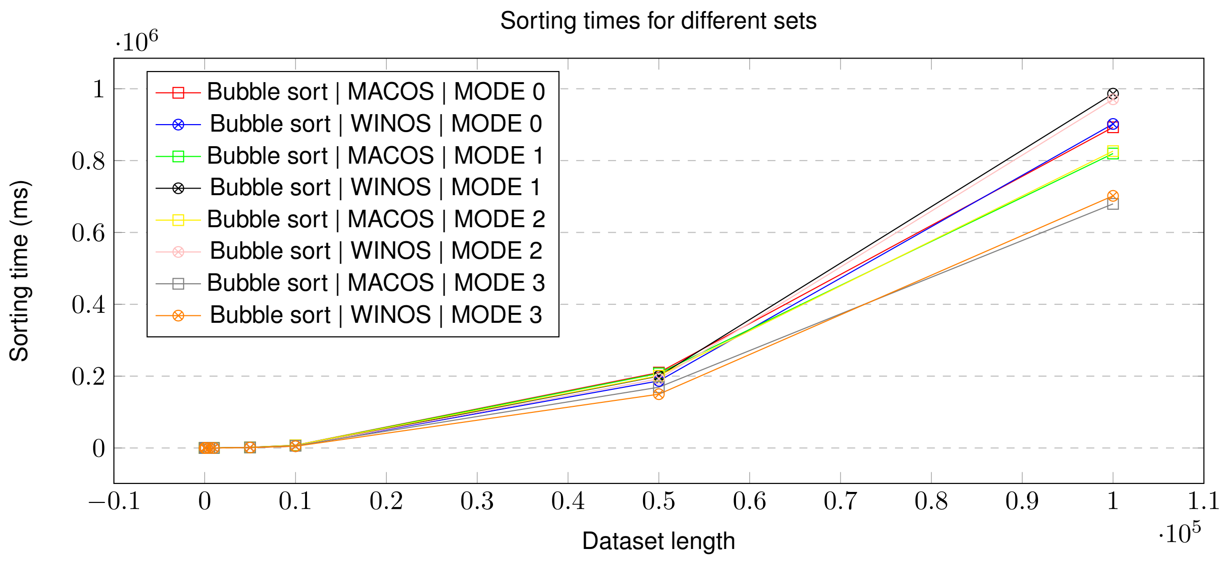
Table 5.2 MACOS specs

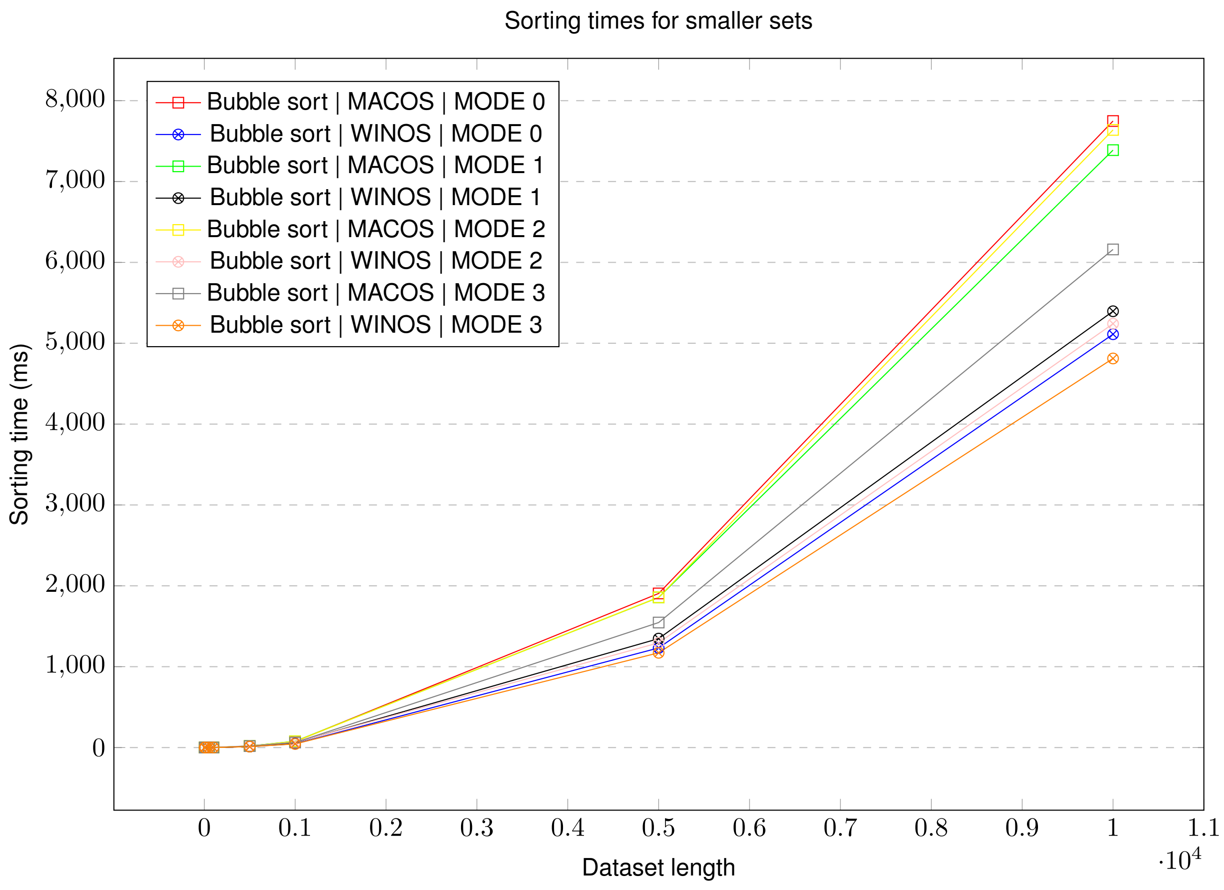
Model	MacBook Air (11-inch, Early 2015)
OS	MacOS Montery 12.6
Processor	1.6 GHz 2-core Intel Core i5
Memory	4 GB 1600 MHz DDR3
Graphics	Intel HD Graphics 6000 1536 MB

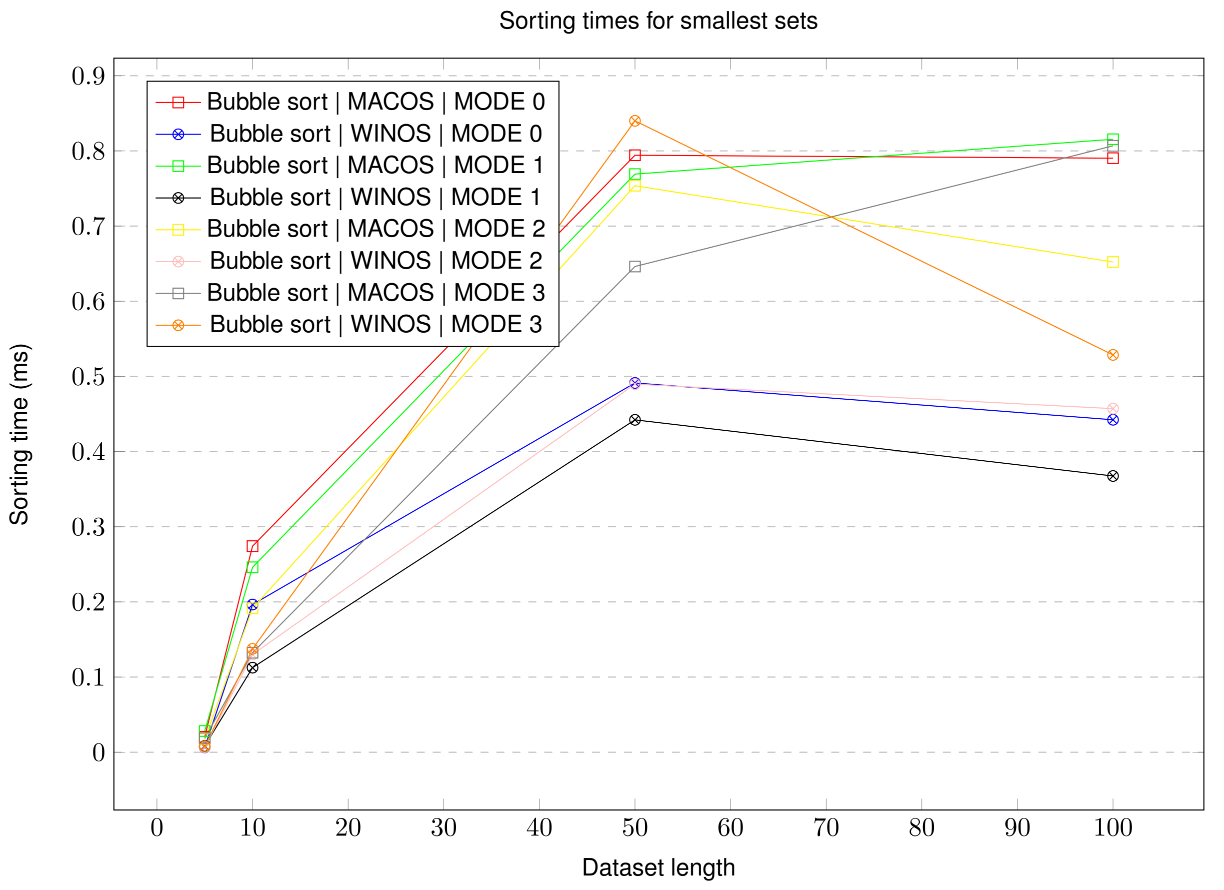
Table 5.3 WIN specs

Model	-
OS	Windows 10 Pro 20H2
Processor	Intel Core i7-8086K @ 4.50 GHz
Memory	80,0 GB 2333 MHz DDR4
Graphics	NVIDIA GeForce GTX 1080

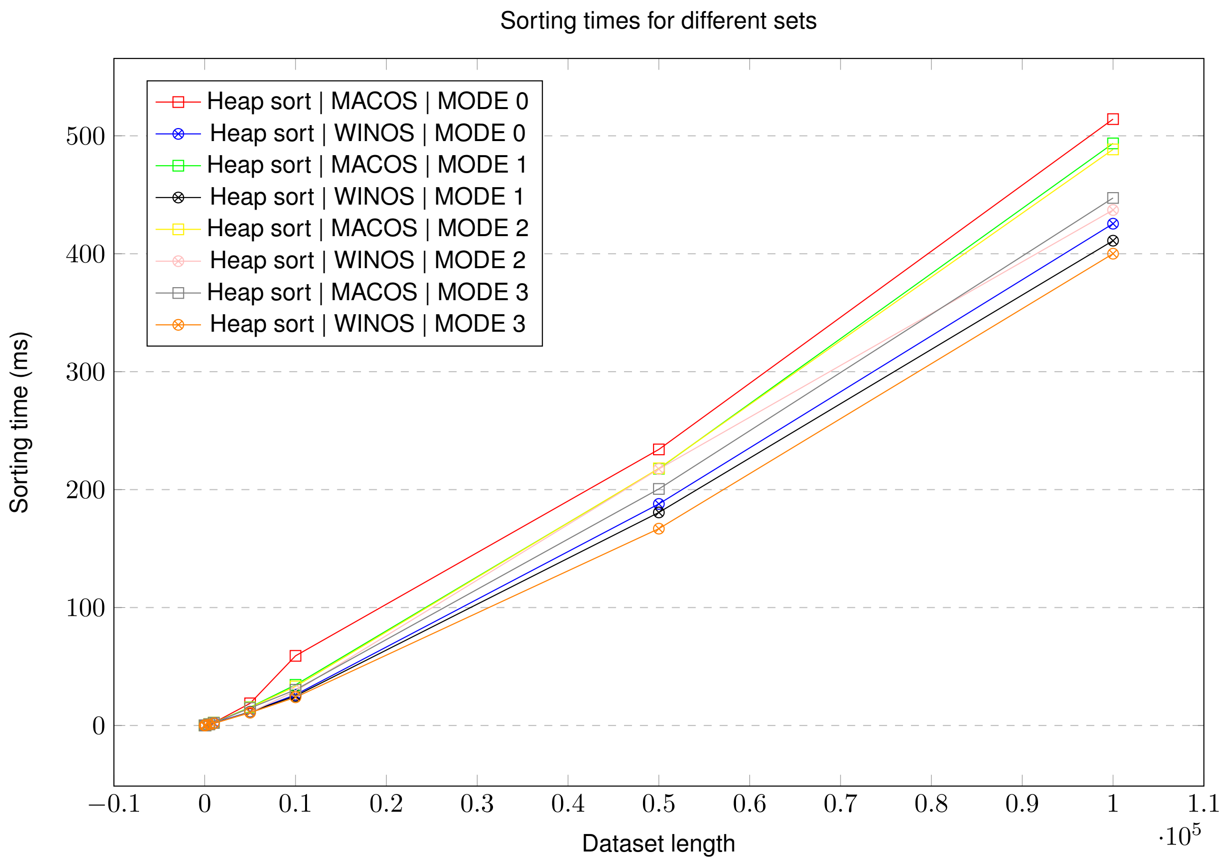
5.0.2 Bubble sorting

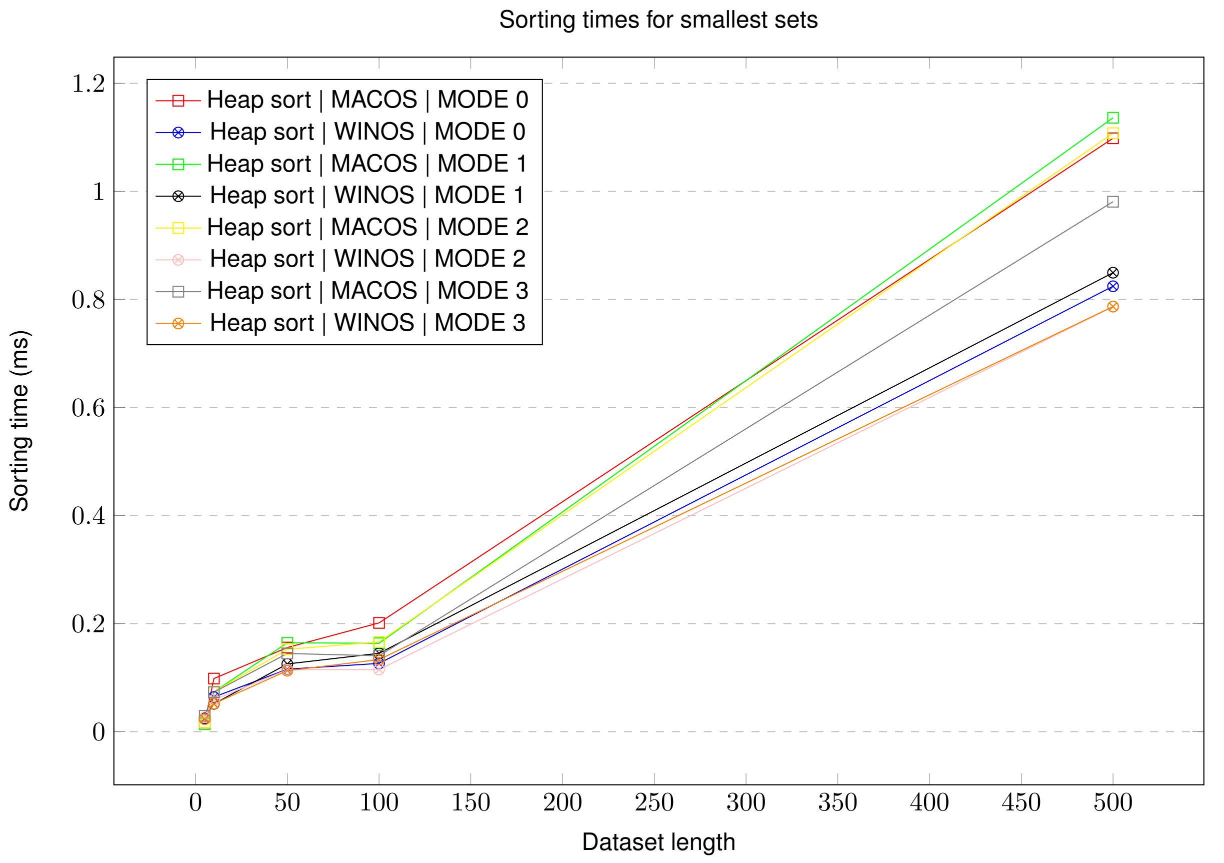



Sorting times for smaller sets
Bubble sort | MACOS | MODE 0
Bubble sort | WINOS | MODE 0
Bubble sort | MACOS | MODE 1
Bubble sort | WINOS | MODE 1
Bubble sort | MACOS | MODE 2
Bubble sort | WINOS | MODE 2
Bubble sort | MACOS | MODE 3
Bubble sort | WINOS | MODE 3
Sorting time (ms)
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
·10^4
Dataset length

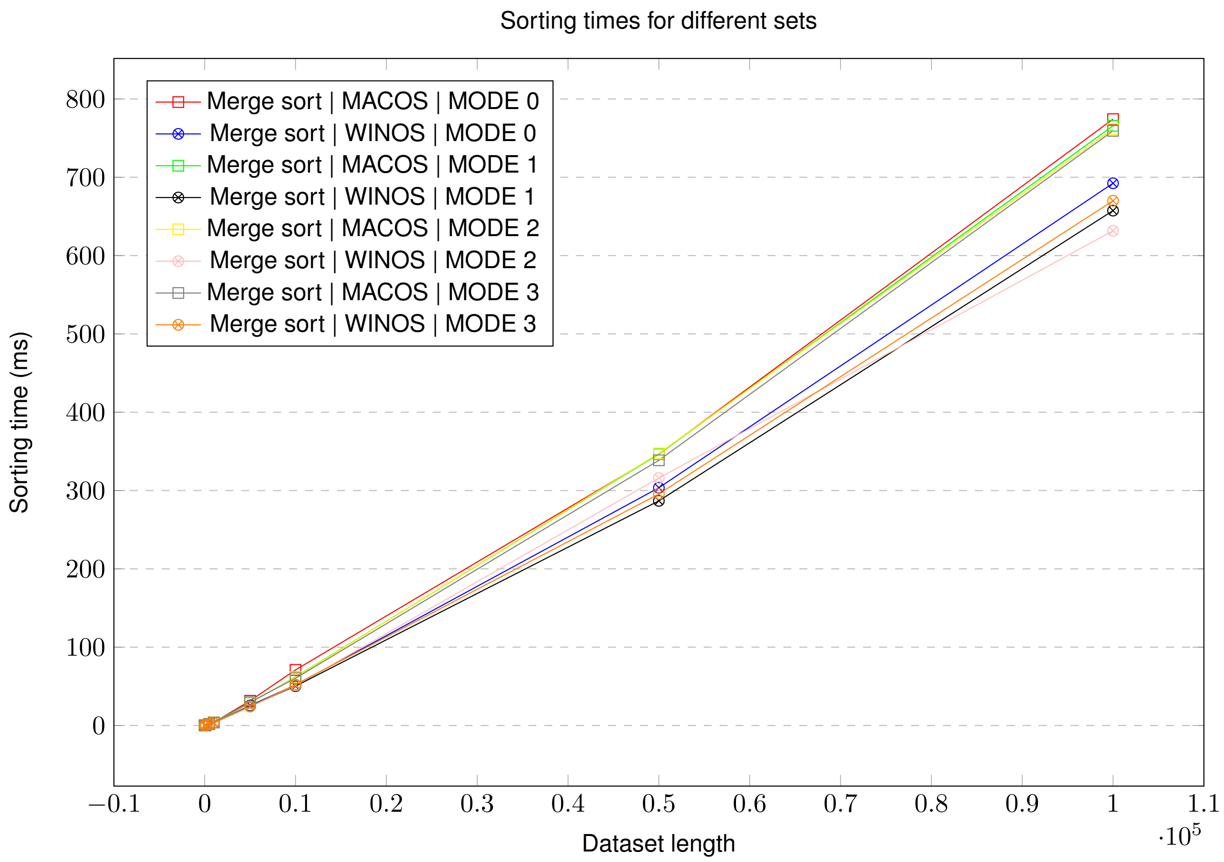

Sorting times for smallest sets
Bubble sort | MACOS | MODE 0
Bubble sort | WINOS | MODE 0
Bubble sort | MACOS | MODE 1
Bubble sort | WINOS | MODE 1
Bubble sort | MACOS | MODE 2
Bubble sort | WINOS | MODE 2
Bubble sort | MACOS | MODE 3
Bubble sort | WINOS | MODE 3
Sorting time (ms)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
80
90
100
Dataset length

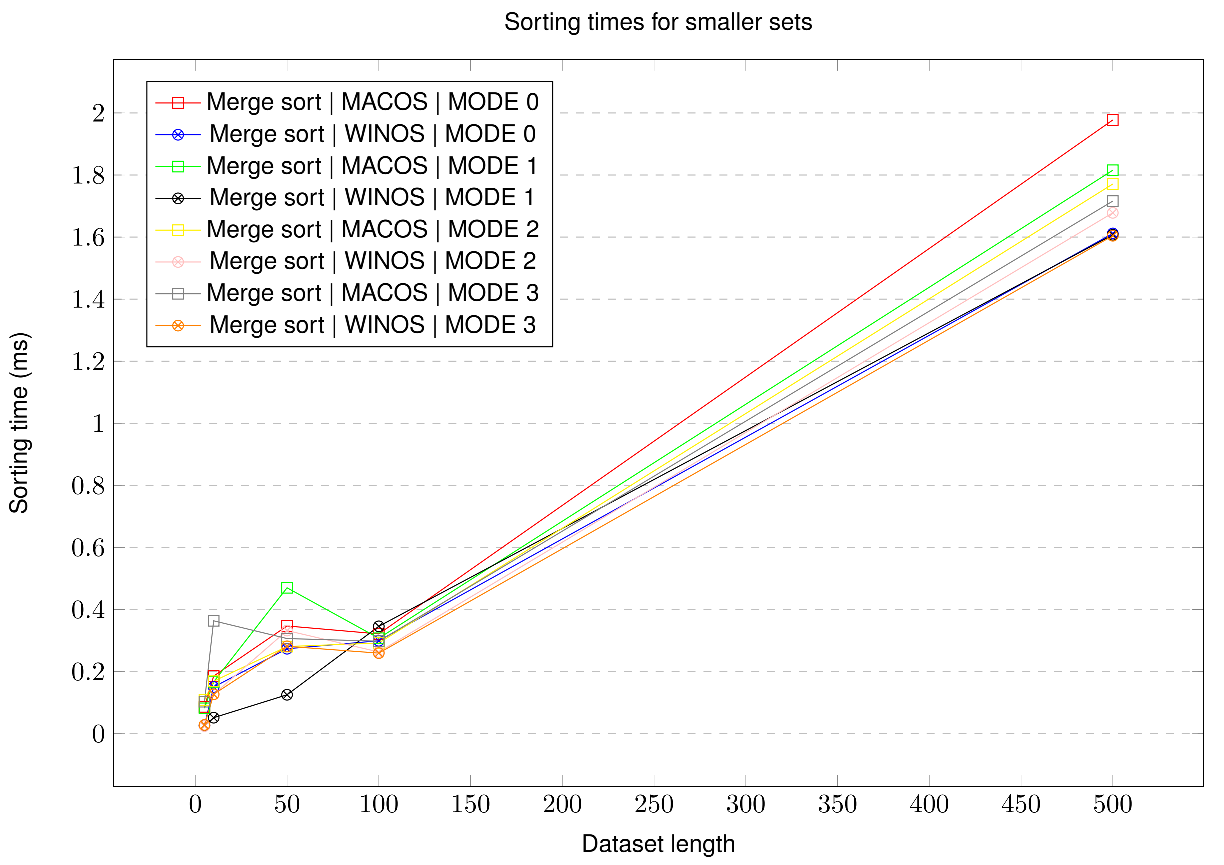
5.0.3 Heap sorting



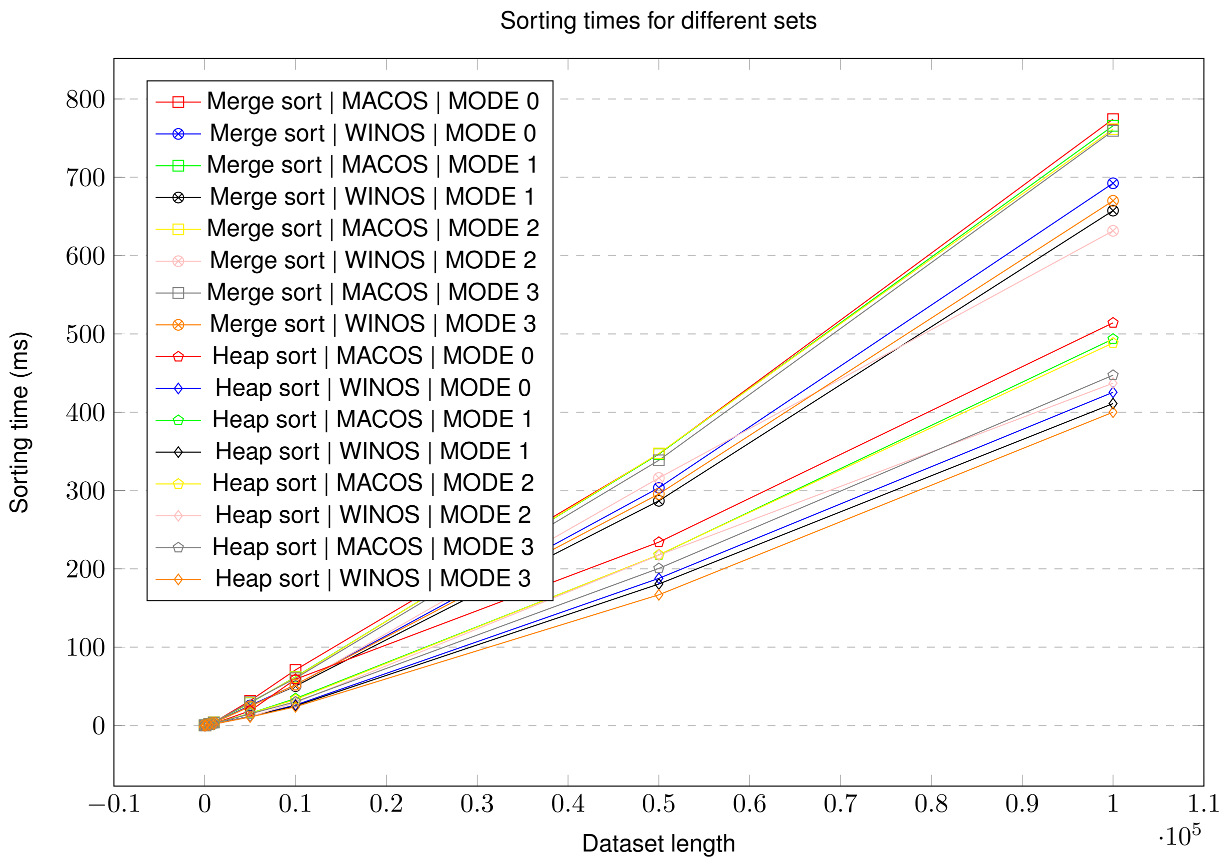


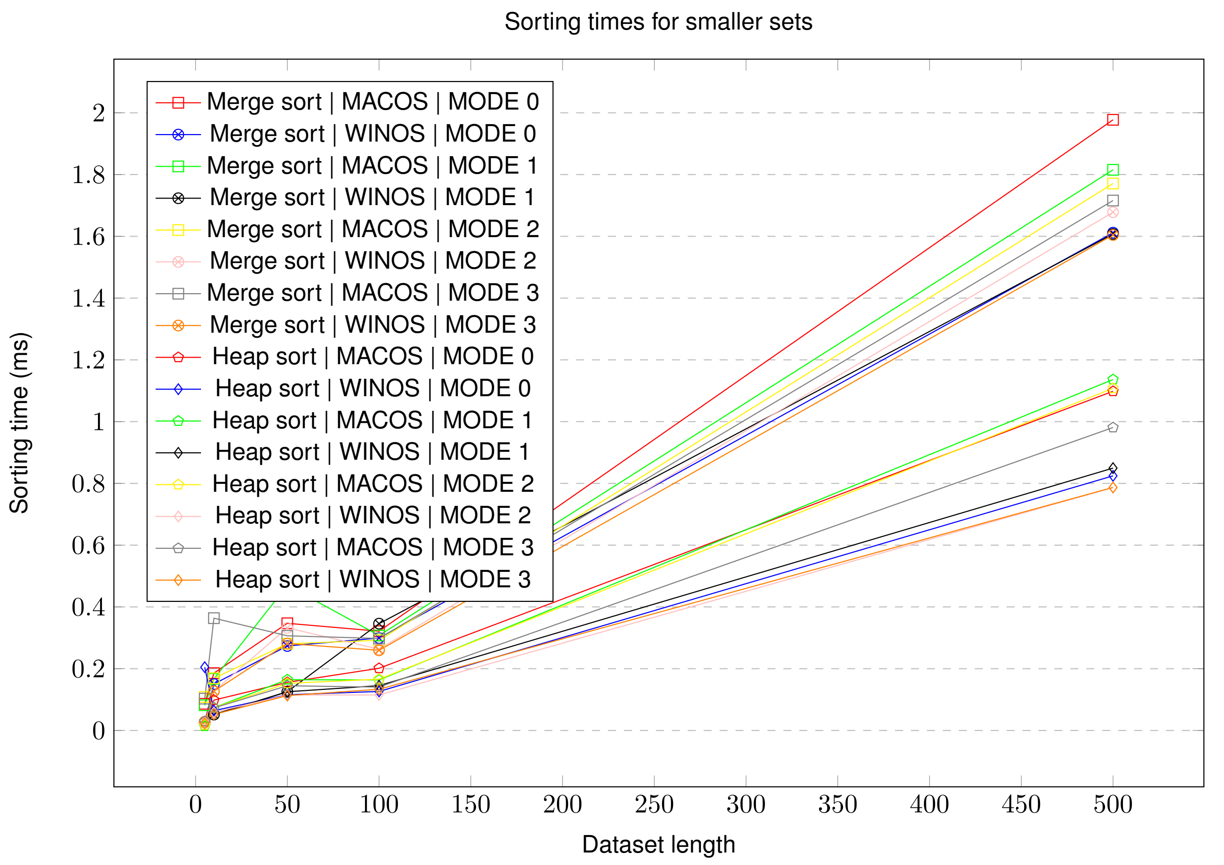
5.0.4 Merge sorting

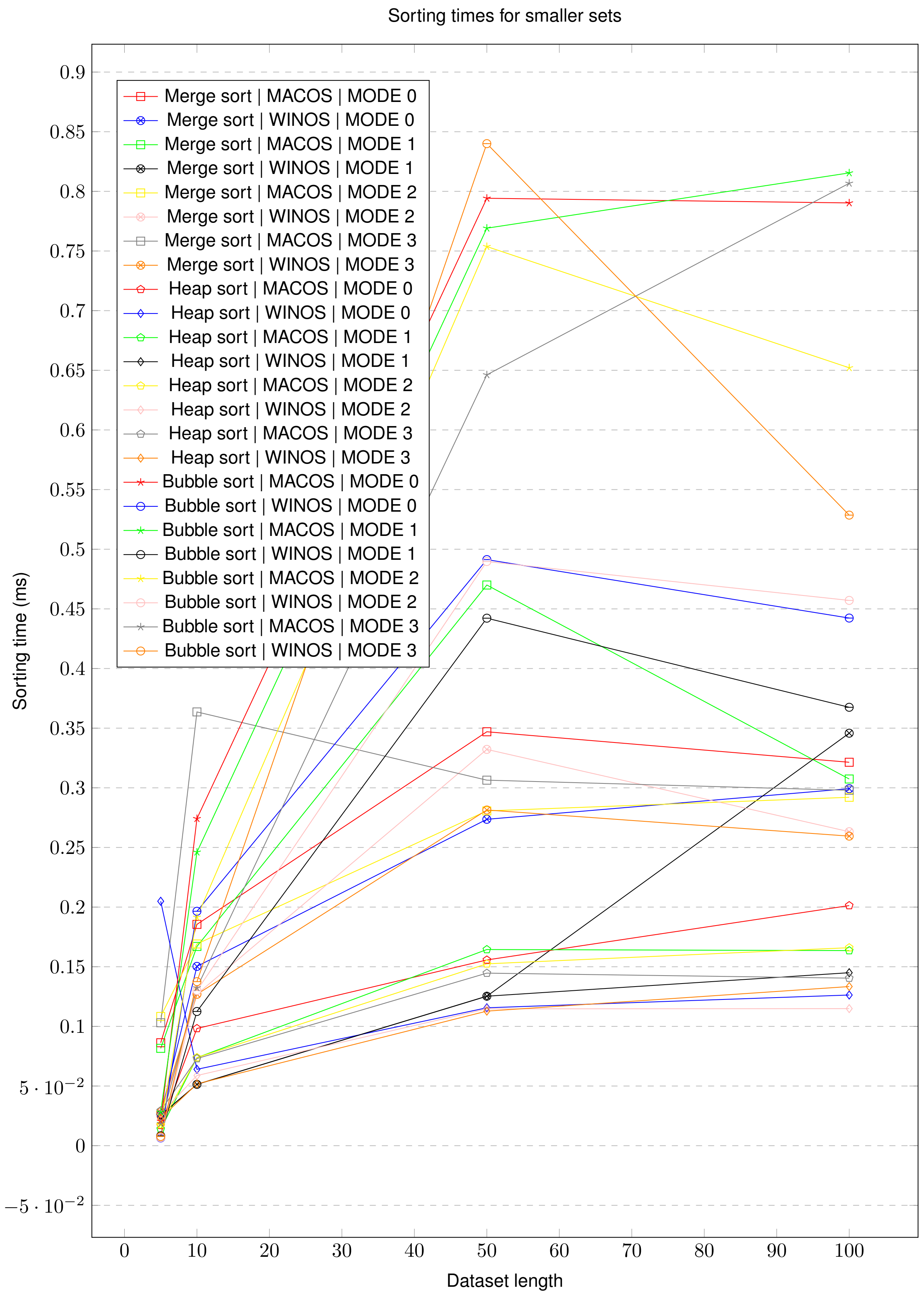




5.0.5 Sorting comparison






Sorting times for smaller sets
Sorting time (ms)
Dataset length
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
$5 \cdot 10^{-2}$
0
$-5 \cdot 10^{-2}$
0
10
20
30
40
50
60
70
80
90
100
Merge sort | MACOS | MODE 0
Merge sort | WINOS | MODE 0
Merge sort | MACOS | MODE 1
Merge sort | WINOS | MODE 1
Merge sort | MACOS | MODE 2
Merge sort | WINOS | MODE 2
Merge sort | MACOS | MODE 3
Merge sort | WINOS | MODE 3
Heap sort | MACOS | MODE 0
Heap sort | WINOS | MODE 0
Heap sort | MACOS | MODE 1
Heap sort | WINOS | MODE 1
Heap sort | MACOS | MODE 2
Heap sort | WINOS | MODE 2
Heap sort | MACOS | MODE 3
Heap sort | WINOS | MODE 3
Bubble sort | MACOS | MODE 0
Bubble sort | WINOS | MODE 0
Bubble sort | MACOS | MODE 1
Bubble sort | WINOS | MODE 1
Bubble sort | MACOS | MODE 2
Bubble sort | WINOS | MODE 2
Bubble sort | MACOS | MODE 3
Bubble sort | WINOS | MODE 3

5.0.6 Conclusion

Verily, as we draw nigh unto the cessation of our discourse, it doth become manifest that Bubble sorting proveth of some worth solely in particular scenarios wherein the tally of elements to be sorted doth not surpass ten. Contrariwise, heap sorting doth exhibit an efficiency and superiority that rendereth it a commendable option for larger datasets.

From a theoretical perspective, Bubble sort possesseth a time complexity of O(n$^2), whichdothentailthatthetimerequiredtosortase$

To elucidate this, let us consider an exemplar wherein we must sort a catalogue of one million numerals. If Bubble sort were utilized, the algorithm would necessitate executing nearly one trillion comparisons to sort the said catalogue. In contradistinction, heap sort would demand only approximately twenty million comparisons to sort the identical catalogue. The discrepancy in computational time requisited to sort the said catalogue using the aforementioned two algorithms is remarkable, and serves to underscore the pre-eminence of heap sorting in terms of computational efficiency.

Moreover, heap sorting's efficiency doth not solely arise from its superior time complexity, but also from its optimal utilization of the computer's memory cache. Unlike Bubble sort, heap sort may operate on subsets of the catalogue, thereby rendering it more suitable for contemporary computer architectures.

In summation, whilst Bubble sorting may have some limited utility, it is widely deemed an antiquated algorithm that is ineffectual for larger datasets. Heap sorting, conversely, is an efficient and sturdy algorithm that executes well on larger datasets by virtue of its optimal use of computer memory. Thus, for most applications, heap sorting remaineth the algorithm of preference for sorting voluminous datasets.

Index