

## Big Oh (Growth Rate)

`make_sale class`

Function: `make_sale::make_sale(QWidget *parent)`

Growth Rate:  $O(1)$

Justification: Only assignment is happening

Function: `make_sale::make_sale(QWidget *parent,  
sales_container* sc,  
Members_Container* mc,  
inventory* i);`

Growth Rate:  $O(1)$

Justification: Only assignment is happening

Function: `make_sale::~~make_sale()`

Growth Rate:  $O(1)$

Justification: Deleting a widget

Function: `void make_sale::printReport()`

Growth Rate:  $O(n)$

Justification: For loop to iterate through n number of sales.

Function: `void make_sale::on_fileInput_clicked()`

Growth Rate:  $O(n)$

Justification: Calls `readFile` which uses a for loop to iterate through all information in txt file

Function: `void make_sale::on_makePurchase_clicked()`

Growth Rate:  $O(n)$

Justification: Calls `print file`, which is  $O(n)$

`sales class`

Function: `sales::sales()`

Growth Rate:  $O(1)$

Justification: Only assignment is happening

Function: `sales::sales(const sales& s)`

Growth Rate:  $O(1)$

Justification: Only assignment happens

Function: `sales::sales(std::string date,  
int id,  
std::string item,  
double price,  
int quantity)`

Growth Rate:  $O(1)$

Justification: Only assignment happens

Function: `sales& sales::operator=(const sales& s)`

Growth Rate  $O(1)$

Justification: Only assignment happens

Function: `bool sales::setItemName(std::string name)`

Growth Rate:  $O(1)$

Justification: Only assignment happens

Function: `bool sales::setDate(std::string date)`

Growth Rate:  $O(n)$

Justification: Calls `std::stoi`, which has a general efficiency of  $O(n)$ <sup>1</sup>

Function: `bool sales::setPrice(double price)`

Growth Rate:  $O(1)$

Justification: Only assignment occurs

Function: `bool sales::setQuantity(int quantity)`

Growth Rate:  $O(1)$

Justification: Only assignment occurs

Function: `bool sales::setId(int id)`

Growth Rate:  $O(1)$

Justification: Only assignment occurs

Function: `std::string sales::getDate() const`

Growth Rate:  $O(1)$

Justification: Only returns an attribute

Function: `int sales::getId() const`

Growth Rate:  $O(1)$

Justification: Only returns an attribute

Function: `std::string sales::getItem() const`

Growth Rate:  $O(1)$

Justification: Only returns an attribute

Function: `double sales::getPrice() const`

Growth Rate:  $O(1)$

Justification: Only returns an attribute

Function: `int sales::getQuantity() const`

---

<sup>1</sup> <https://www.cplusplus.com/reference/string/stoi/>

Growth Rate:  $O(1)$

Justification: Only returns an attribute

Function: `double sales::getRevenue() const`

Growth Rate:  $O(1)$

Justification: Only returns an attribute

Function: `bool sales::operator==(const sales& s) const`

Growth Rate:  $O(1)$

Justification: Only compares values using `==`

`sales_container` Class

Function: `sales_container::sales_container()`

Growth Rate:  $O(1)$

Justification: Only assignment happens

Function: `sales_container::sales_container(unsigned int size)`

Growth Rate:  $O(1)$

Justification: Only assignment happens

Function: `sales_container::sales_container(unsigned int size,  
const sales& initial)`

Growth Rate:  $O(n)$

Justification: A for loop is used to iterate through and assign  $n$  elements from parameter `initial`

Function: `sales_container::sales_container(const sales_container& s)`

Growth Rate:  $O(n)$

Justification: A for loop is used to iterate through and assign  $n$  elements from parameter `s`

Function: `sales_container::~~sales_container()`

Growth Rate:  $O(n)$

Justification: Deleting an array

Function: `unsigned int sales_container::capacity() const`

Growth Rate:  $O(1)$

Justification: Only returning an attribute

Function: `unsigned int sales_container::size() const`

Growth Rate:  $O(1)$

Justification: Only returning an attribute

Function: `bool sales_container::empty() const`

Growth Rate:  $O(1)$

Justification: Only comparing an attribute to `0`

Function: `sales& sales_container::operator[](unsigned int index) const`

Growth Rate:  $O(1)$

Justification: Only returning an attribute

Function: `double sales_container::getTotalRevenue() const`

Growth Rate:  $O(n)$

Justification: Uses a for loop to iterate through  $n$  elements in the sales container

Function: `int sales_container::find(const sales& s) const`

Growth Rate:  $O(n)$

Justification: Uses a for loop to iterate through  $n$  elements in the sales container

Function: `int sales_container::find(std::string name) const`

Growth Rate:  $O(n)$

Justification: Uses a for loop to iterate through  $n$  elements in the sales container

Function: `int sales_container::find(int id) const`

Growth Rate:  $O(n)$

Justification: Uses a for loop to iterate through  $n$  elements in the sales container

Function: `int sales_container::getItemQuantity(std::string item) const`

Growth Rate:  $O(n)$

Justification: Uses a for loop to iterate through  $n$  elements in the sales container

Function: `bool sales_container::contains(sales &s) const`

Growth Rate:  $O(n)$

Justification: Uses a for loop to iterate through  $n$  elements in the sales container

Function: `bool sales_container::contains(int id) const`

Growth Rate:  $O(n)$

Justification: Uses a for loop to iterate through  $n$  elements in the sales container

Function: `bool sales_container::outFile(std::string output) const`

Growth Rate:  $O(n)$

Justification: Uses a for loop to iterate through  $n$  elements in the sales container

Function: `void sales_container::push_back(const sales &value)`

Growth Rate:  $O(1)$

Justification: Only appends a value to the container

Function: `void sales_container::push_back(QWidget* parent,  
 const sales& value,  
 inventory& inventory,  
 Members_Container& all_members)`

Growth Rate:  $O(n)$

Justification: Calls `inventory.search` which is  $O(n)$

Function: `void sales_container::pop_back()`

Growth Rate:  $O(1)$

Justification: Decrements attribute by 1

Function: `void sales_container::erase(const sales& s)`

Growth Rate:  $O(n)$

Justification: Calls `find`, which is  $O(n)$  and uses a for loop to shift  $n$ -index elements.

Function: `void sales_container::set_size(int size)`

Growth Rate:  $O(1)$

Justification: Only assigns an attribute from the parameter

Function: `void sales_container::set_capacity(int capacity)`

Growth Rate:  $O(1)$

Justification: Only assigns an attribute from parameter

Function: `sales_container& sales_container::operator=(const sales_container& s)`

Growth Rate:  $O(n)$

Justification: Uses a for loop to iterate through  $n$  elements of the parameter

Function: `void sales_container::reserve(unsigned int capacity)`

Growth Rate:  $O(n)$

Justification: Calls function `copy()` which is  $O(n)$

Function: `void sales_container::resize(unsigned int size)`

Growth Rate:  $O(n)$

Justification: Calls function `reserve()` which is  $O(n)$

Function: `void sales_container::clear()`

Growth Rate:  $O(n)$

Justification: Only assigns `0` to attributes

Function: `bool sales_container::readFile(QWidget* parent,  
std::string input,  
inventory& inventory,  
Members_Container& members)`

Growth Rate:  $O(n)$

Justification: Uses a while loop to iterate through  $n$  lines from a txt file

Function: `bool sales_container::readFile(std::string name)`

Growth Rate:  $O(n)$

Justification: Uses a while loop to iterate through  $n$  lines of a txt file

`daily_sales` Class

Function: `daily_sales::daily_sales(QWidget *parent)`

Growth Rate:  $O(1)$

Justification: Only assigns attributes

Function: `daily_sales::daily_sales(QWidget *parent, sales_container* sc, Members_Container* mc)`

Growth Rate:  $O(n)$

Justification: Calls assignment operator for sales and Members containers, which is  $O(n)$

Function: `daily_sales::~~daily_sales()`

Growth Rate:  $O(1)$

Justification: Deletes Widget

Function: `void daily_sales::on_submit_clicked()`

Growth Rate:  $O(n)$

Justification: Calls `generate_daily_sales()` which is  $O(n)$

Function: `void daily_sales::generate_daily_sales(std::string date, int flag)`

Growth Rate:  $O(n^2)$

Justification: 4 different non-nested for loops each call a function that is  $O(n)$ .

yearly\_sales Class

Function: `yearly_sales::yearly_sales(QWidget *parent)`

Growth Rate:  $O(1)$

Justification: Only assigns attributes

Function: `yearly_sales::yearly_sales(QWidget *parent,  
sales_container* all_sales,  
Members_Container* mc,  
inventory* iv)`

Growth Rate:  $O(n)$

Justification: Calls assignment operators which are all  $O(n)$  complexity

Function: `yearly_sales::~~yearly_sales()`

Growth Rate:  $O(1)$

Justification: Deletes ui attribute

Function: `void yearly_sales::clearInput()`

Growth Rate: unknown

Justification: calls clear which is a member function of QTextEdit, complexity unknown

Function: `void yearly_sales::on_submit_clicked()`

Growth Rate:  $O(n^2)$

Justification: Uses 8 different non-nested for loops to generate report, worst case scenario a for loop calls a function that is also  $O(n)$

inventory Class

Function: `inventory::inventory()`

Growth Rate:  $O(1)$

Justification: Only initializes attributes

Function: `inventory::inventory(int size)`

Growth Rate:  $O(1)$

Justification: Only initializes attributes

Function: `inventory::inventory(int size, const Item& it)`

Growth Rate:  $O(n)$

Justification: Uses a for loop to iterate through n elements given by size

Function: `inventory::inventory(const inventory& i)`

Growth Rate:  $O(n)$

Justification: Uses a for loop to iterate through n elements of i

Function: `inventory::~~inventory()`

Growth Rate:  $O(n)$

Justification: Calls delete on an array

Function: `int inventory::capacity() const`

Growth Rate:  $O(1)$

Justification: Returns the value of an attribute

Function: `int inventory::size() const`

Growth Rate:  $O(1)$

Justification: Returns the value of an attribute

Function: `bool inventory::empty() const`

Growth Rate:  $O(1)$

Justification: Compares an attribute to 0

Function: `bool inventory::contains(std::string name) const`

Growth Rate:  $O(n)$

Justification: Uses a for loop to iterate through  $n$  elements of inventory

Function: `void inventory::push_back(const Item &it)`

Growth Rate:  $O(1)$

Justification: This function calls `reallocate` which is  $O(n)$

Function `void inventory::pop_back()`

Growth Rate:  $O(1)$

Justification: This function decrements an attribute by 1

Function: `int inventory::search(const Item& t) const`

Growth Rate:  $O(n)$

Justification: This function uses a for loop to iterate through  $n$  elements of inventory

Function: `int inventory::search(std::string name)`

Growth Rate:  $O(n)$

Justification: This function uses a for loop to iterate through  $n$  elements of inventory

Function: `void inventory::remove(const Item &it)`

Growth Rate:  $O(n)$

Justification: This function calls `search` which is  $O(n)$  and uses a for loop to shift  $n$ -index items in the inventory

Function: `void inventory::reallocate(int cap)`

Growth Rate:  $O(n)$

Justification: This function uses `copy`, which we assume is  $O(n)^2$

Function: `void inventory::resize(int size)`

Growth Rate:  $O(n)$

Justification: This function calls `copy`

Function: `void inventory::clear()`

Growth Rate:  $O(n)$

Justification: This function call `delete` on an array, which we assume is  $O(n)$

Function: `void inventory::set_size(int size)`

Growth Rate:  $O(1)$

Justification: This function assigns a value to an attribute

---

<sup>2</sup> <http://cplusplus.com/reference/algorithm/copy/>

Function: void inventory::set\_capacity(int cap)  
Growth Rate:  $O(1)$   
Justification: This function assigns a value to an attribute

Function: Item& inventory::operator [](int index) const  
Growth Rate:  $O(1)$   
Justification: This function only returns an attribute

Function: inventory& inventory::operator+=(const Item &item)  
Growth Rate:  $O(n)$   
Justification: This function calls push\_back which is  $O(n)$

Function: inventory& inventory::operator=(const inventory &it)  
Growth Rate:  $O(n)$   
Justification: This function uses a for loop to iterate through n elements of inventory

Function: void inventory::readFile(std::string input)  
Growth Rate:  $O(n)$   
Justification: This function uses a while loop to iterate through n lines of a txt file

Function: void inventory::outFile(std::string name)  
Growth Rate:  $O(n)$   
Justification: This function uses a for loop to iterate through n elements of inventory

## InventoryTracker Class

Function: InventoryTracker::InventoryTracker(QWidget \*parent)  
Growth Rate:  $O(1)$   
Justification: We assume that the function setupUi() is  $O(1)$

Function: InventoryTracker::InventoryTracker(QWidget \*parent, inventory\* iv, sales\_container\* sales)  
Growth Rate:  $O(1)$   
Justification: This function assigns values to the attributes

Function: InventoryTracker::~~InventoryTracker()  
Growth Rate:  $O(1)$   
Justification: We assume that delete ui is  $O(1)$

Function: void InventoryTracker::on\_exit\_clicked()  
Growth Rate:  $O(1)$   
Justification: This function calls the destructor which is  $O(1)$  and we assume that close() is also  $O(1)$

Function: void InventoryTracker::empty()  
Growth Rate:  $O(1)$   
Justification: This function calls setText() which we assume is  $O(1)$

Function: void InventoryTracker::generate\_inventory\_list()



Growth Rate:  $O(n^2)$

Justification: This function uses 2 different for loops that each make a call to `sales_container::find()` which is  $O(n)$

Function: `Item::Item()`

Growth Rate:  $O(1)$

Justification: This function assigns values to the attributes

Function: `Item::Item(int id, int quantity, double price)`

Growth Rate:  $O(1)$

Justification: This function assigns values to the attributes

Function: `Item::Item(string n, int q, double p)`

Growth Rate:  $O(1)$

Justification: This function assigns values to the attributes

Function: `void Item::set_item_number(int id)`

Growth Rate:  $O(1)$

Justification: This function assigns a value to an attribute

Function: `void Item::set_quantity(int quan)`

Growth Rate:  $O(1)$

Justification: This function assigns a value to an attribute

Function: `void Item::set_price(double p)`

Growth Rate:  $O(1)$

Justification: This function assigns a value to an attribute

Function: `int Item::get_quantity() const`

Growth Rate:  $O(1)$

Justification: This function returns the value of an attribute

Function: `double Item::get_price() const`

Growth Rate:  $O(1)$

Justification: This function returns the value of an attribute

Function: `int Item::get_ID() const`

Growth Rate:  $O(1)$

Justification: This function returns the value of an attribute

Function: `double Item::get_total() const`

Growth Rate:  $O(1)$

Justification: This function does a basic calculation and returns its value

Function: `string Item::get_item_name() const`

Growth Rate:  $O(1)$

Justification: This function returns the value of an attribute

Function: `bool operator ==(const Item& i1, const Item& i2)`

Growth Rate:  $O(1)$

Justification: This function compares two items

Function: `item_reports::item_reports(QWidget *parent)`

Growth Rate:  $O(1)$

Justification: This assumes that `setupUi()` is  $O(1)$

Function: `item_reports::item_reports(QWidget *parent,  
sales_container* all_sales,  
inventory* all_items)`

Growth Rate:  $O(1)$

Justification: This assumes that `setupUi()` is  $O(1)$

Function: `item_reports::~~item_reports()`

Growth Rate:  $O(1)$

Justification: This assumes that `delete ui` is  $O(1)$ !

Function: `void item_reports::on_submitButton_clicked()`

Growth Rate:  $O(n)$

Justification: This function calls `allItemReport` or `singleItemReport` which are both  $O(n)$

Function: `void item_reports::singleItemReport(std::string itemName)`

Growth Rate:  $O(n)$

Justification: This function uses 2 different non-nested for-loops to iterate through  $n$  elements of a sales container

Function: `allItemReport()`

Growth Rate:  $O(n^2)$

Justification: This function has 2 different non-nested for loops that each call `sales_container::find()` which is  $O(n)$

MainWindow Class

Function: `MainWindow::MainWindow(QWidget *parent)`

Growth Rate:  $O(1)$

Justification: This assumes that `setupUi()` is  $O(1)$

Function: `MainWindow::~~MainWindow()`

Growth Rate:  $O(n)$

Justification: This function calls `outFile` from the inventory, sales and member containers which are all  $O(n)$

Function: `void MainWindow::on_quitButton_clicked()`

Growth Rate:  $O(n)$

Justification: This function calls `outFile` from the inventory, sales and member containers which are all  $O(n)$

Function: `void MainWindow::on_dailySales_clicked()`

Growth Rate:  $O(n^2)$

Justification: This function calls the `dailySales` class which has a function with  $O(n^2)$

Function: `void MainWindow::on_manageMembers_clicked()`

Growth Rate:  $O(n^2)$

Justification: This function calls the manageMembers class which has a function with  $O(n^2)$

Function: void MainWindow::on\_makeSale\_clicked()

Growth Rate:  $O(n^2)$

Justification: This function calls the make\_sale class which has a function with  $O(n^2)$

Function: void MainWindow::on\_yearlySales\_clicked()

Growth Rate:  $O(n^2)$

Justification: This function calls the yearly\_sales class which has a function with  $O(n^2)$

Function: void MainWindow::on\_manageInventory\_clicked() Growth Rate:  $O(n^2)$

Justification: This function calls the manage\_inventory class which has a function with  $O(n^2)$

Function: void MainWindow::on\_itemReport\_clicked()

Growth Rate:  $O(n^2)$

Justification: This function calls the item\_report class which has a function with  $O(n^2)$

Function: void MainWindow::on\_purchaseReports\_clicked()

Growth Rate:  $O(n^2)$

Justification: This function calls the purchase\_reports class which has a function with  $O(n^2)$

manageMembers Class

Function: manageMembers::manageMembers(QWidget \*parent)

Growth Rate:  $O(1)$

Justification: This assumes that setupUi() is  $O(1)$

Function: manageMembers::manageMembers(QWidget \*parent, Members\_Container\* mc,  
sales\_container\* sc)

Growth Rate:  $O(n)$

Justification: This function calls the assignment operator for Members\_Container which is  $O(n)$

Function: manageMembers::~~manageMembers()

Growth Rate:  $O(1)$

Justification: This assumes that delete ui is  $O(1)$

Function: void manageMembers::on\_button\_addMember\_clicked()

Growth Rate:  $O(1)$

Justification: This assumes that hiding or showing an input or button is  $O(1)$

Function: void manageMembers::on\_submit\_clicked()

Growth Rate:  $O(n)$

Justification: This function calls Members\_Container::add\_member() which is  $O(n)$

Function: void manageMembers::on\_membersFromFile\_clicked()

Growth Rate:  $O(1)$

Justification: This assumes that hiding input fields and buttons is  $O(1)$

Function: `void manageMembers::on_submitFile_clicked()`

Growth Rate:  $O(n)$

Justification: This function calls `add_bulk_members` which is  $O(n)$

Function: `void manageMembers::on_button_delete_Member_clicked()`

Growth Rate:  $O(1)$

Justification: This function assumes that hiding and showing fields of a widget is  $O(1)$

Function: `void manageMembers::on_submitDelete_clicked()`

Growth Rate:  $O(n)$

Justification: This function calls `remove_member()` which is  $O(n)$

Function: `void manageMembers::on_viewMemberInfo_clicked()`

Growth Rate:  $O(1)$

Justification: This function assumes that hiding and showing fields of a widget is  $O(1)$

Function: `void manageMembers::on_displayButton_clicked()`

Growth Rate:  $O(n)$

Justification: This function calls `Members_Container::contains()` and `Members_Container::get_member()` which are both  $O(n)$

Function: `void manageMembers::on_membersConvToBasic_clicked()`

Growth Rate:  $O(n^2)$

Justification: This function has a for loop that calls `Members_Container::add_member()` which is  $O(n)$

Function: `void manageMembers::on_button_renew_membership_clicked()`

Growth Rate:  $O(1)$

Justification: This assumes that showing and hiding attributes of a widget is  $O(1)$

Function: `void manageMembers::on_submitRenew_clicked()`

Growth Rate:  $O(n^2)$

Justification: This function uses a for loop that calls `Member_Container::contains()` which is  $O(n)$

Function: `void manageMembers::on_membershipExpirations_clicked()`

Growth Rate:  $O(1)$

Justification: This assumes that showing and hiding attributes of a widget is  $O(1)$

Function: `void manageMembers::on_submitDate_clicked()`

Growth Rate:  $O(n^2)$

Justification: This function has a for loop that calls `_get_member()` which is  $O(n)$

## Member Class

Function: `Member::Member(std::string _name, int _membership_number, bool _premium_member, std::string _membership_expiration)`

Growth Rate:  $O(1)$

Justification: This function only assigns values to the attributes

Function: `Member::Member()`

Growth Rate:  $O(1)$

Justification: This function only assigns values to the attributes

Function: `Member::Member(std::string name)`

Growth Rate:  $O(1)$

Justification: This function only assigns values to the attributes

Function: `void Member::set_name(const std::string& _name)`

Growth Rate:  $O(1)$

Justification: This function only assigns values to the attributes

Function: `void Member::set_membership_number()`

Growth Rate:  $O(1)$

Justification: This function only assigns values to the attributes

Function: `void Member::upgrade_member(const std::string& todays_date)`

Growth Rate:  $O(1)$

Justification: Does not use any loops nor calls a function that uses a loop

Function: `Member::extend_membership()`

Growth Rate:  $O(1)$

Justification: Does not use any loops nor calls a function that uses a loop

Function: `std::string getInfo()`

Growth Rate:  $O(1)$

Justification: Returns a value

## memberPurchase Class

Function: `memberPurchase::memberPurchase(QWidget *parent)`

Growth Rate:  $O(1)$

Justification: This assumes that `setupUi()` is  $O(1)$

Function: `memberPurchase::memberPurchase(QWidget *parent,  
sales_container* sc,  
Members_Container* mc,  
inventory* iv)`

Growth Rate:  $O(1)$

Justification: This assumes that `setUpUi()` is  $O(1)$

Function: `memberPurchase::~~memberPurchase()`

Growth Rate:  $O(1)$

Justification: This assumes that `delete ui` is  $O(1)$

Function: `void memberPurchase::on_submit_clicked()`

Growth Rate:  $O(n^2)$

Justification: This function calls `allMemberReport()` which is  $O(n^2)$

Function: `void memberPurchase::allMemberReport()`

Growth Rate:  $O(n^2)$

Justification: This function uses a for loop that calls `Members_Container::get_member()` which is  $O(n)$

Members\_Container Class

Function: `Members_Container::Members_Container()`

Growth Rate:  $O(1)$

Justification: This assumes that creating a new Member is  $O(1)$

Function: `Members_Container::~~Members_Container()`

Growth Rate:  $O(n)$

Justification: This assumes that deleting an array is  $O(n)$

Function: `bool Members_Container::contains(const int& _membership_number)`

Growth Rate:  $O(n)$

Justification: This function uses a for loop to iterate through  $n$  elements in the container

Function: `bool Members_Container::contains(const std::string& _name)`

Growth Rate:  $O(n)$

Justification: This function uses a for loop to iterate through  $n$  elements in the container

Function: `void Members_Container::add_member(const Member &new_member)`

Growth Rate:  $O(n)$

Justification: This function uses a for loop to iterate through  $n$  elements in the container

Function: void Members\_Container::remove\_member(const std::string &\_name)

Growth Rate:  $O(n^2)$

Justification: This element uses a nested for loop

Function: void Members\_Container::remove\_member(const int& \_membership\_number)

Growth Rate:  $O(n^2)$

Note: This function uses a nested for loop

Function: void Members\_Container::upgrade\_membership(const std::string& \_name, const std::string& \_date)

Growth Rate:  $O(n)$

Justification: This function uses a for loop to iterate through n elements in the container

Function: void Members\_Container::upgrade\_membership(const int &\_membership\_number, const std::string& \_date)

Growth Rate:  $O(n)$

Justification: This function uses a for loop to iterate through n elements in the container

Function: void Members\_Container::add\_bulk\_members(const std::string& file\_location)

Growth Rate:  $O(n^2)$

Justification: This function uses a while loop that calls add\_member() which is  $O(n)$

Function: bool Members\_Container::outFile(std::string output)

Growth Rate:  $O(n)$

Justification: This function uses a for loop to iterate through n elements in the container

Function: bool Members\_Container::validateMemberFile(std::string file)

Growth Rate:  $O(n)$

Justification: This function uses a while loop to iterate through the txt file