COMP 3000 Semester Project Analysis:

Chemical Database

Graham Oliver

Table of Contents:

chemicalDatabase::showList

Code:

```
void chemicalDatabase::showList(){
    cout << "All Chemicals in Database:" << endl;
    for(int i=0; i<=chems.size()-1; i++){
        cout << chems[i].name << endl;
    }
    cout << endl;
}
```

Analysis:

All cases are the same for this function. There is one for loop dependent on input size that grows in computation time linearly.

Big O:

Best: O(n)

Average: O(n)

Worst: O(n)

chemicalDatabase::getProperty

Code:

```
void chemicalDatabase::getProperty(){
    string chemical;
    int again = 1;
    int index = 0;
    int match = 0;
    int type;
    do{
        cout << "Enter the name of the chemical you would like to find properties
for." << endl;
        cin >> chemical;
        for(int i=0; i<=chems.size()-1; i++){
            if(chemical == chems[i].name){
                match = 1;
                again = 0;
                index = i;
            }
        }
        if(match == 0){
            cout << "No chemical with the name " << chemical << " could be
found." << endl << "Would you like to try again?" << endl
            << "Enter 0 for No and 1 for Yes." << endl;
                do{
                    cin >> again;
                    if(again<0 | again>1){
                        cout << "Enter 0 for No and 1 for Yes." << endl;
                    }
                }
                while(again<0 | again>1);
        }
    }
    while(again == 1);
    if(match == 1){
        cout << "Property Menu:" << endl << "Enter 0 to show the description for
" << chemical << "." << endl
        << "Enter 1 to show the molar mass." << endl << "Enter 2 to show the
melting point." << endl
        << "Enter 3 to show the boiling point." << endl << "Enter 4 to show the
critical temperature." << endl
        << "Enter 5 to show the critical pressure." << endl << "Enter 6 to show
all information." << endl;
        cin >> type;
        switch(type){
```

```
            case 0:
                cout << chems[index].description << endl;
                break;
            case 1:
                cout << "Molar Mass: " << chems[index].properties[0] << " g/mol"
<< endl;
                break;
            case 2:
                cout << "Melting Point: " << chems[index].properties[1] << " K"
<< endl;
                break;
            case 3:
                cout << "Boiling Point: " << chems[index].properties[2] << " K"
<< endl;
            case 4:
                cout << "Critical Temperature: " << chems[index].properties[3] <<
" K" << endl;
                break;
            case 5:
                cout << "Critical Pressure: " << chems[index].properties[4] << "
bar" << endl;
                break;
            case 6:
                cout<< chems[index].description << endl << endl << "Molar Mass: "
<< chems[index].properties[0] << " g/mol" << endl
                << "Melting Point: " << chems[index].properties[1] << " K" <<
endl << "Boiling Point: " << chems[index].properties[2] << " K" << endl
                << "Critical Temperature: " << chems[index].properties[3] << " K"
<< endl << "Critical Pressure: " << chems[index].properties[4] << " bar" << endl;
                break;
        }
        cout << endl;
    }
}
```

Analysis:

The best-case scenario is for the initial entered name to match one in the database and the properties can be displayed. This requires the value to be calculated through an if within a for within a while. The worst-case scenario is that the value entered does not match one in the database. This would require going through an if within a for within an if within a while.

Big O:

Best case: O(n^3)

Worst Case: O(n^4)

Average: O(n^3)

chemicalDatabase::vaporPressure:

Code:

```
void chemicalDatabase::vaporPressure(){
    double Pstar;
    string chemical;
    int again = 1;
    int index = 0;
    int match = 0;
    double temp;
    do{
        cout << "Enter the name of the chemical you would like to find the vapor
pressure of." << endl;
        cin >> chemical;
        for(int i=0; i<=chems.size()-1; i++){
            if(chemical == chems[i].name){
                match = 1;
                index = i;
                again = 0;
            }
        }
        if(match == 0){
            cout << "No chemical with the name " << chemical << " could be
found." << endl << "Would you like to try again?" << endl
            << "Enter 0 for No and 1 for Yes." << endl;
                do{
                    cin >> again;
                    if(again<0 | again>1){
                        cout << "Enter 0 for No and 1 for Yes." << endl;
                    }
                }
                while(again<0 | again>1);
        }
    }
    while(again == 1);
    if(match == 1){
        double A = chems[index].antoine[0]; double B = chems[index].antoine[1];
double C = chems[index].antoine[2];
        cout << "Enter the temperature in Kelvin at which you would like to find
the vapor pressure of " << chemical << "." << endl;
        cin >>  temp;
        Pstar = pow(10,A-(B/(temp+C)));
        cout << "Vapor Pressure of " << chemical << " at " << temp << " K: "<<
endl << Pstar << " bar" << endl << endl;
    }
```

Analysis:

The best case scenario is if the name entered matches the name of a database chemical, in this case the value is calculated through an if within a for within a while. The worst case is if the name entered does not match, in this case the value is calculated through an if within a while within a while.

Big O:

Best case: O(n^3)

Worst case: O(n^4)

Average: O(n^3)

chemicalDatabase::addremove:

Code:

```
void chemicalDatabase::getProperty(){
    string chemical;
    int again = 1;
    int index = 0;
    int match = 0;
    int type;
    do{
        cout << "Enter the name of the chemical you would like to find properties
for." << endl;
        cin >> chemical;
        for(int i=0; i<=chems.size()-1; i++){
            if(chemical == chems[i].name){
                match = 1;
                again = 0;
                index = i;
            }
        }
        if(match == 0){
            cout << "No chemical with the name " << chemical << " could be
found." << endl << "Would you like to try again?" << endl
            << "Enter 0 for No and 1 for Yes." << endl;
                do{
                    cin >> again;
                    if(again<0 | again>1){
                        cout << "Enter 0 for No and 1 for Yes." << endl;
                    }
                }
                while(again<0 | again>1);
        }
    }
    while(again == 1);
    if(match == 1){
        cout << "Property Menu:" << endl << "Enter 0 to show the description for
" << chemical << "." << endl
        << "Enter 1 to show the molar mass." << endl << "Enter 2 to show the
melting point." << endl
        << "Enter 3 to show the boiling point." << endl << "Enter 4 to show the
critical temperature." << endl
        << "Enter 5 to show the critical pressure." << endl << "Enter 6 to show
all information." << endl;
        cin >> type;
        switch(type){
```

```
            case 0:
                cout << chems[index].description << endl;
                break;
            case 1:
                cout << "Molar Mass: " << chems[index].properties[0] << " g/mol"
<< endl;
                break;
            case 2:
                cout << "Melting Point: " << chems[index].properties[1] << " K"
<< endl;
                break;
            case 3:
                cout << "Boiling Point: " << chems[index].properties[2] << " K"
<< endl;
            case 4:
                cout << "Critical Temperature: " << chems[index].properties[3] <<
" K" << endl;
                break;
            case 5:
                cout << "Critical Pressure: " << chems[index].properties[4] << "
bar" << endl;
                break;
            case 6:
                cout<< chems[index].description << endl << endl << "Molar Mass: "
<< chems[index].properties[0] << " g/mol" << endl
                << "Melting Point: " << chems[index].properties[1] << " K" <<
endl << "Boiling Point: " << chems[index].properties[2] << " K" << endl
                << "Critical Temperature: " << chems[index].properties[3] << " K"
<< endl << "Critical Pressure: " << chems[index].properties[4] << " bar" << endl;
                break;
        }
        cout << endl;
    }
}
```

Analysis:

The worst-case scenario is that the name entered does not match one from the database. This requires the value to be calculated through an if within a while within an if within a while. For the best case scenario, when the name entered matches, the value must be calculated through an if within a for within a while.

Big O:

Best case: O(n^3)

Worst case: O(n^4)

Average: O(n^3)

chemicalDatabase::stockAccess(string chemical):

Code:

```
double chemicalDatabase::stockAccess(string chemical){
    int index = 0;
    int match = 0;
    for(int i=0; i<=chems.size()-1; i++){
        if(chemical == chems[i].name){
            match = 1;
            index = i;
        }
    }
    if(match == 1){
        return chems[index].properties[0];
    }
    else{
        return 0;
    }
}
```

Analysis:

There is no best or worst case scenario, every value must be calculated through an if within a for.

Big O: Best = Worst = Average: O(n^2)

stockRoom::showStock():

Code:

```
void stockRoom::showStock(){
    if(stocks.size()<1){
        cout << "Stockroom is empty, add stock of a chemical first." << endl;
    }
    else{
        cout << "Current Stock:" << endl;
        for(int i=0; i<=stocks.size()-1; i++){
            cout << stocks[i].name << ":    " << stocks[i].amountkg << "
kg        " << stocks[i].amountmoles << " moles" << endl;
        }
        cout << endl;
    }
}
```

Analysis:

Base case is the stockroom is empty, in this case the value must be calculated through one if statement. Worst case is that the stockroom is not empty, in which case the value must be calculated through a for within an if/else.

Big O:

Best case: O(n)

Worst Case: O(n^2)

Average: O(n^2)

stockRoom::addremove(chemicalDatabase &d):

Code:

```
void stockRoom::addremove(chemicalDatabase &d){
    int again = 1;
    string chemical;
    double mm = 0;
    double newkg;
    double kgadd;
    double newmol;
    int index = -1;
    do{
        cout << "Enter the name of the chemical of which you would like to add or
remove stock." << endl;
        cin >> chemical;
        mm = d.stockAccess(chemical);
        if(mm != 0){
            again = 0;
        }
        else{
            cout << "The chemical you add or remove stock to must be in the
database first." << endl;
        }
    }
    while(again == 1);
    int already = 0;
    if(stocks.size()>=1){
        for(int i=0; i<=stocks.size()-1; i++){
            if(stocks[i].name == chemical)
            already = 1;
            index = i;
        }
        if(stocks[0].name == chemical){
            already = 1;
            index = 0;
        }
    }
    cout << "How much of " << chemical << " would you like to add or
remove.\nEnter a value in kilograms, use a negative number to remove stock" <<
endl;
    cin >> kgadd;
    if(already == 0){
        stock Snew;
        Snew.name = chemical;
        Snew.amountkg = kgadd; Snew.amountmoles = kgadd*1000/mm;
```

```
        stocks.push_back(Snew);
        index = stocks.size()-1;
    }
    else{
        newkg = kgadd + stocks[index].amountkg;
    newmol = kgadd*1000/mm + stocks[index].amountmoles;
        stocks[index].amountkg = newkg;
        stocks[index].amountmoles = newmol;
    }
    if (stocks[index].amountkg <= 0){
        int remove = 0;
        cout << "There is no more " << chemical << " in stock. Would you like to
remove it from the stock room?" << endl
        << "Enter 0 for No and 1 for Yes." << endl;
        cin >> remove;
        if(remove == 1){
            if(stocks.size()==1){
                stocks.clear();
            }
            else{
                for(int k=index; k<=stocks.size()-2; k++){
                    stocks[k] = stocks[k+1];
            }
                stocks.pop_back();
                cout << chemical << " has been removed from the stockroom." <<
endl;
            }
        }
        else{
            stocks[index].amountkg = 0;
            stocks[index].amountmoles = 0;
        }
    }
    else{
    cout << "You now have " << stocks[index].amountkg << " kg, or " <<
stocks[index].amountmoles << " moles of " << chemical << " left." << endl <<
endl;
    }
}
```

Analysis: Best case is if the chemical entered and changed matches one already in stock and the remaining stock is more than zero. In this case the value must be sent through an if within a for within an if. The worst case scenario is that all of a stock is removed and the chemical is removed from the

stockroom, and there is more than one chemicals in the stockroom. In this case the value must be sent through a for within an if/else within an if/else, within an if.

Big O:

Best: O(n^3)

Worst: O(n^4)

Average: O(n^3)

Results:

Class	Function	Best	Average	Worst
chemicalDatabase	showList	O(n)	O(n)	O(n)
chemicalDatabase	getProperty	O(n^3)	O(n^3)	O(n^4)
chemicalDatabase	vaporPressure	O(n^3)	O(n^3)	O(n^4)
chemicalDatabase	stockAcess	O(n^2)	O(n^2)	O(n^2)
chemicalDatabase	addremove	O(n^3)	O(n^3)	O(n^4)
stockRoom	showStock	O(n)	O(n^2)	O(n^2)
stockRoom	addremove	O(n^3)	O(n^3)	O(n^4)

Table 1: Big O Values for Program Functions