

|  |  |  |  |
| --- | --- | --- | --- |
| Course Code | : | DMT211 | |
| Course Name | : | Algorithm Analysis and Design | |
| Lecturer | : | Geetha Kanaparan | |
| Academic Session | : | 2021/09 | |
| Assessment Title | : | Assignment Question 2 | |
| Submission Due Date | : | 10/12/2021 | |
| Prepared by |  |  | |
| : | Student ID | Student Name |
|  | DMT2002036 | Yim Jing Xiang |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Date Received | : |  | |
|  |  |  | |
| Feedback from Lecturer:  Mark: | | | |

**Own Work Declaration**

I/We hereby understand my/our work would be checked for plagiarism or other misconduct, and the softcopy would be saved for future comparison(s).

I/We hereby confirm that all the references or sources of citations have been correctly listed or presented and I/we clearly understand the serious consequence caused by any intentional or unintentional misconduct.

This work is not made on any work of other students (past or present), and it has not been submitted to any other courses or institutions before.



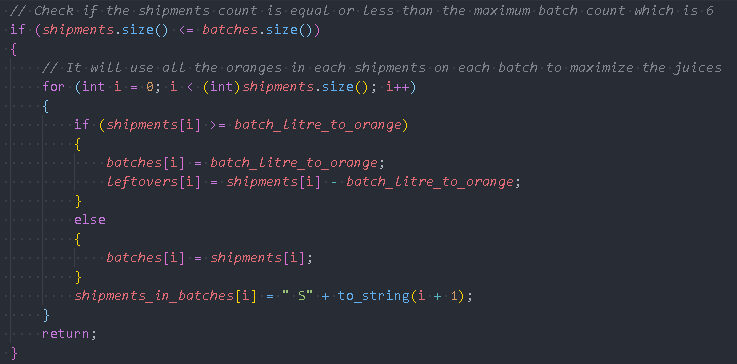
Signature: Yim Jing Xiang

Date: 09/12/2021

**Yim Jing Xiang DMT2002036**

For the conversion from oranges to litres, I have used 14 oranges = 1 litre of orange juice. (*How many oranges does it take to produce 1 liter of orange juice?*)

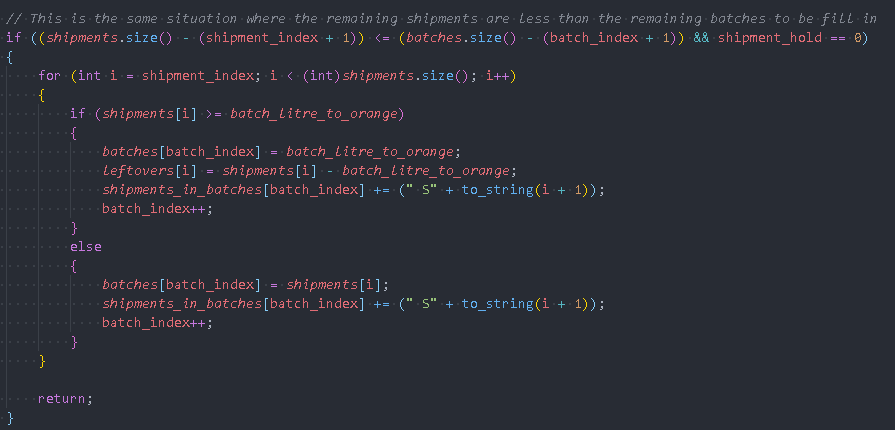
**Question c**

First, we will let n be the sample size, which is the number of shipments.

In the first if loop in the algorithm, this will check if the number of shipments is equal or less than the maximum number of batches which is 6. If it meets the requirements, it will run the for loop for n times then it will return to main(). This will give us O(n) in this part.



And for the while loop itself, although it looks like it depends on the shipments size and batches size, but we know that if it runs the while loop, the shipments size will always be larger than batches size at the first place.



For the first if loop in the while loop, we have a for loop inside it which runs for (shipments size – shipment\_index) times. As this if loop is to process the situation where the remaining shipments are less than the remaining batches to be fill in, it will run (n – shipment\_index) times, where shipment\_index points to the current shipment we are processing. If there are 15 shipments, and we already ran through 11 shipments, and there are 4 batches left. In this situation, we will run 4 times in this for loop and then return to the main(). 11+4 = 15 which is the number of shipments.



We will let m be the number of shipments processed before it hits the if loop where it processes the situation where the remaining shipments are less than the remaining batches to be fill in.

In the next if else loop, it handles the addition of oranges in shipments to fulfil the maximum output to each batch. We can see that it runs for m times.

By combining both of the if else loop in the while loop, it will give us O(n).

And in conclusion, if it hits the first if loop, it will never hit the while loop. If it didn’t hit the first if loop, it will then hit the while loop. So, in the end, the algorithm’s time complexity is O(n). It depends on the number of shipments.

**Question d**

**1.** In the first set of data, it will be {1, 1, 1, 14, 14, 14}.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| My Algorithm | 1 | 1 | 1 | 14 | 14 | 14 |
| Other Algorithm | 1+1+1+14 | 14 | 14 |  |  |  |

My algorithm’s first part is to check whether the number of shipments is equal or lesser than the number of batches, which is 6. If it meets the requirement, it will then throw in the oranges in each shipment into each batch.

In my algorithm, it will have 45 used oranges and 0 discards, which means it produced the maximum number of juices that all the shipments can.

In other algorithm, they might not check for this situation and will just do accumulation of oranges until it hit the maximum juices can be produced in the batch, then only proceed to next batch. It will have 42 used oranges and 3 discarded oranges.

**2.** The second set of data will be {5, 5, 1, 14, 2, 14, 6, 14}.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| My Algorithm | 5+5+1 | 14 | 2 | 14 | 6 | 14 |
| Other Algorithm | 5 | 5 | 14 | 14 | 6 | 14 |

In this situation, my algorithm will do accumulation until it hits a condition where the remaining shipments equal the remaining batches. Then it will do the same thing as in case one, where it will throw the remaining shipments into each remaining batches. It will have 61 used oranges and 0 discarded oranges.

In other algorithm, they might pick the first 6 largest elements in the shipments. It will have 58 used oranges and 3 discarded oranges.

**3.** The third set of data will be {1, 18, 5, 19, 2, 14, 16, 4}

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| My Algorithm | 1+18 | 5+19 | 2 | 14 | 16 | 4 |
| Other Algorithm | 1 | 18 | 5 | 19 | 2 | 14 |

In this situation, the other algorithm might not do accumulations of oranges from each shipment. This will make the other algorithm have 59 used oranges and 20 discarded oranges.

For my algorithm, it will accumulate and check for the number of remaining shipments and the number of remaining batches. It will have 64 used oranges and 15 discarded oranges, which makes my algorithm more efficient.

**4.** The fourth set of data will be {1, 2, 10, 12, 14, 14, 14, 14}

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| My Algorithm | 1+2+10 | 12 | 14 | 14 | 14 | 14 |
| Other Algorithm | 1+2+10+12 | 14 | 14 | 14 | 14 |  |

For my algorithm, although it has skipped through the first check of the number of shipments and the number of batches, there is still some passive checking in the while loop. After the accumulations of oranges from S1, S2 and S3, it will just stop accumulating the oranges and throw in oranges from remaining shipments into remaining batches. It will have 81 used oranges and 0 discarded oranges.

In the other algorithm which doesn’t implement this checking, it will waste a lot of oranges. It will have 70 used oranges and 11 discarded oranges.

**5.** The fifth set of data will be {1…n}, where n = number of shipments && n >= 14(6d) = 64d && all shipments have at least 1 orange. My algorithm will always get the maximum juices produced which is 6d litres. In other algorithm, they might have to do several checking to make sure they don’t discard some oranges. This is where the time complexity of O(n) of my algorithm comes in handy which is optimal for running time and also optimises the total number of juices produced in total of 6 batches.

**Appendix**

#include <iostream>

#include <string>

#include <vector>

using namespace std;

*// Input Validation*

bool input\_validation(string *input*, bool *flag*)

{

    for (int i = 0; i < (int)*input*.length(); i++)

    {

        if (isdigit(*input*[i]) == false && *input* != "-1")

        {

            return false;

        }

    }

    if (stoi(*input*) < -1 && *flag* == false)

    {

        return false;

    }

    if (stoi(*input*) < 0 && *flag* == true)

    {

        return false;

    }

    return true;

}

*// The shipment processing functino*

void process(vector<int> *shipments*, vector<int> &*batches*, vector<int> &*leftovers*, int *batch\_litre\_to\_orange*, vector<string> &*shipments\_in\_batches*)

{

    int shipment\_index = 0, batch\_index = 0;

*// Check if the shipments count is equal or less than the maximum batch count which is 6*

    if (*shipments*.size() <= *batches*.size())

    {

*// It will use all the oranges in each shipments on each batch to maximize the juices*

        for (int i = 0; i < (int)*shipments*.size(); i++)

        {

            if (*shipments*[i] >= *batch\_litre\_to\_orange*)

            {

*batches*[i] = *batch\_litre\_to\_orange*;

*leftovers*[i] = *shipments*[i] - *batch\_litre\_to\_orange*;

            }

            else

            {

*batches*[i] = *shipments*[i];

            }

*shipments\_in\_batches*[i] = " S" + to\_string(i + 1);

        }

        return;

    }

    int shipment\_hold = 0;

*// If the shipments count is more than 6, then it will run the below while loop*

    while (shipment\_index < (int)*shipments*.size() && (batch\_index + 1) <= (int)*batches*.size())

    {

*// This is the same situation where the remaining shipments are less than the remaining batches to be fill in*

        if ((*shipments*.size() - (shipment\_index + 1)) <= (*batches*.size() - (batch\_index + 1)) && shipment\_hold == 0)

        {

            for (int i = shipment\_index; i < (int)*shipments*.size(); i++)

            {

                if (*shipments*[i] >= *batch\_litre\_to\_orange*)

                {

*batches*[batch\_index] = *batch\_litre\_to\_orange*;

*leftovers*[i] = *shipments*[i] - *batch\_litre\_to\_orange*;

*shipments\_in\_batches*[batch\_index] += (" S" + to\_string(i + 1));

                    batch\_index++;

                }

                else

                {

*batches*[batch\_index] = *shipments*[i];

*shipments\_in\_batches*[batch\_index] += (" S" + to\_string(i + 1));

                    batch\_index++;

                }

            }

            return;

        }

*// This handles the additions of oranges in shipments to fullfil the maximum output to each batches*

        if ((*shipments*[shipment\_index] + shipment\_hold) >= *batch\_litre\_to\_orange*)

        {

*batches*[batch\_index] = *batch\_litre\_to\_orange*;

*leftovers*[shipment\_index] = *shipments*[shipment\_index] - (*batch\_litre\_to\_orange* - shipment\_hold);

*shipments\_in\_batches*[batch\_index] += (" S" + to\_string(shipment\_index + 1));

            shipment\_index++;

            batch\_index++;

            shipment\_hold = 0;

        }

        else

        {

            if ((*shipments*.size() - (shipment\_index + 1)) <= (*batches*.size() - (batch\_index + 1)))

            {

*batches*[batch\_index] = *shipments*[shipment\_index] + shipment\_hold;

*shipments\_in\_batches*[batch\_index] += (" S" + to\_string(shipment\_index + 1));

                shipment\_index++;

                batch\_index++;

                shipment\_hold = 0;

            }

            else

            {

                shipment\_hold += *shipments*[shipment\_index];

*shipments\_in\_batches*[batch\_index] += (" S" + to\_string(shipment\_index + 1));

                shipment\_index++;

            }

        }

    }

    return;

}

int main()

{

    string input, batch\_litre;

    int increment = 0, batch\_litre\_to\_oranges;

    bool inputFlag = false;

    vector<int> shipments, batches(6, 0);

    vector<string> shipments\_in\_batches(6, "");

*// User input of amount of oranges in shipments*

    while (input != "-1")

    {

        cout << "Enter the number of oranges in shipment " << increment + 1 << endl;

        cout << "(To next step, please enter -1)\n";

        cin >> input;

        while (!input\_validation(input, inputFlag))

        {

            system("cls");

            cout << "Invalid input.\n"

                 << "Please enter the number of orange in shipment " << increment + 1 << endl;

            cin >> input;

        }

        increment++;

        system("cls");

        if (input != "-1")

        {

            shipments.push\_back(stoi(input));

        }

    }

    inputFlag = true;

*// User input of amount of orange juice to be produced in each batch*

    cout << "Enter the amount of orange juice to be produced in each batch => ";

    cin >> batch\_litre;

    while (!input\_validation(batch\_litre, inputFlag))

    {

        system("cls");

        cout << "Invalid input.\n"

             << "Please enter the amount of orange juice to be produced in each batch.\n";

        cin >> input;

    }

    system("cls");

*// Output of the shipments that the user has input*

    cout << "These are the shipments and amount of orange juice to be produced.\n\_\_\_\_\_\_\_\_\_\n";

    for (int i = 0; i < (int)shipments.size(); i++)

    {

        cout << "S" << i + 1 << " | " << shipments[i] << endl;

    }

    cout << "\_\_\_\_\_\_\_\_\_\n\n";

*// Output of the amount of orange juice to be produced that the user has input*

    cout << "Amount of orange juice to be produced = " << batch\_litre << " litres.\n";

    vector<int> leftovers(shipments.size(), 0);

    batch\_litre\_to\_oranges = 14 \* stoi(batch\_litre);

    system("pause");

    system("cls");

*// Call the shipments processing function*

    process(shipments, batches, leftovers, batch\_litre\_to\_oranges, shipments\_in\_batches);

    float total = 0;

*// This is the output of orange juice produced in each batch*

    cout << "The batches \n";

    for (int i = 0; i < (int)batches.size(); i++)

    {

        cout << "B" << i + 1 << " = " << batches[i] / (float)14 << " litres\n";

        total += batches[i] / (float)14;

    }

    cout << "Total = " << total << " litres\n";

    cout << endl;

*// This is the output of corresponding shipment of oranges used in the batch*

    cout << "Shipments used in batches \n";

    for (int i = 0; i < (int)shipments\_in\_batches.size(); i++)

    {

        cout << "Shipments In Batches " << i + 1 << " = " << shipments\_in\_batches[i] << endl;

    }

    cout << endl;

*// This is the output of discarded oranges from each shipment*

    cout << "The discarded oranges from each shipment \n";

    for (int i = 0; i < (int)leftovers.size(); i++)

    {

        cout << "Leftover" << i + 1 << " = " << leftovers[i] << " oranges\n";

    }

    cout << endl;

    return 0;

}

**Reference**

IAC Publishing. (n.d.). *How many oranges does it take to produce 1 liter of orange juice?* Reference. Retrieved December 9, 2021, from https://www.reference.com/world-view/many-oranges-produce-1-liter-orange-juice-17d6687d558f8f44.