CSC 212 : Group Assignment

Place: University of the Western Cape

Lecturer: HLONIPHANI CLASSACK MALULEKE

Date : 23 October 2020

Group Name: TrenTech

Author list

Casey (3857065)

Douw (3924812)

Josh (3851192)

Keanu (3851727)

Titania (3722490)

Yusuf (3970279)

Table of contents

Table of contents 1

Introduction 2

Social Network Graph from Articles 2

- **Methods** 2
- **Results** 7
- **Discussion** 8

Bus services (A knapsack problem) 9

- **Methods** 9
- **Results** 14
- **Discussion** 15

References 16

Introduction

In this study two proposed questions need to be answered: 1. What does a disperse web of politicians, companies, places and celebrities look like? 2. How would a bus company load passengers with their luggage to maximise a single trip profit? Therefore, the objective of this Group assignment is to find the answer to the proposed questions.Furthermore, the report contains two main sections: Social Network Graph of Articles and bus services (A knapsack problem). In addition, these sections report on the respective methods, results and, a discussion of the methods and results to report the information used and answer the two proposed questions.

Social Network Graph from Articles

Methods

In this section a suitable method: a table of entities, is provided to create an understanding of the disperse web of politicians, companies, places and celebrities from the articles provided in the study. Nonetheless, justification for this method is also provided below.

SPACE LEFT BLANK INTENTIONALLY

A Table of **people inside articles**

Articles	ID	Entity (surname of person)	label
1	1	Nsthalintshali	Person
1	2	Mashilo	Person
3	3	Yolisa Matakata	Person
3	4	Noloyiso Rwexana	Person
3	5	Rape victim #1	Person
3	6	Rape victim #2	Person
3	7	Rape suspect #1	Person
3	8	Rape suspect #2	Person
3	9	Convicted rapist	Person
3	10	Rwexana	Person
3	11	Matakata	Person
4	12	Van der Sandt	Person
4	13	Asulnura	Person
4	14	Kolawole	Person
4	15	Ngamije	Person
4	16	Malec	Person
4	17	Onyago	Person
4	18	Edoro-Glines	Person
4	19	Murua	Person
4	20	Edozien	Person
4	21	Gevisser	Person
4	22	Khan	Person

***Table 1 on page 2** : A Table representing a list of people mentioned in the four articles provided.Created in Google drive.*

Table 1 was used to organize the person entities and their relationships, captured from the four articles. Furthermore this is a suitable method as it gives a clear illustration of all the person entities in the four articles. Furthermore the id assigned to each person makes it easier to reference the row of data, and allow retrieval of this row of data to be processed into meaningful information.

Table representing **links between person entities**

Article	From (ID number)	To (ID number)	Type of relationship
1	1	2	friends
1	2	1	friends
3	3	4;5;6;7;8;9;10;11	friends
3	4	3;5;6;7;8;9;10;11	friends
3	5	3;4;6;7;8;9;10;11	friends
3	6	3;4;5;7;8;9;10;11	friends
3	7	3;4;5;6;8;9;10;11	friends
3	8	3;4;5;6;7;9;10;11	friends
3	9	3;4;5;6;7;8;10;11	friends
3	10	3;4;5;6;7;8;9;11	friends
3	11	3;4;5;6;7;8;9;10	friends
4	12	13;14;15;16;17;18;19;20;21;22	friends
4	13	12;14;15;16;17;18;19;20;21;22	friends
4	14	12;13;15;16;17;18;19;20;21;22	friends
4	15	12;13;14;16;17;18;19;20;21;22	friends
4	16	12;13;14;15;17;18;19;20;21;22	friends
4	17	12;13;14;15;16;18;19;20;21;22	friends
4	18	12;13;14;15;16;17;19;20;21;22	friends
4	19	12;13;14;15;16;17;18;20;21;22	friends
4	20	12;13;14;15;16;17;18;19;21;22	friends
4	21	12;13;14;15;16;17;18;19;20;22	friends
4	22	12;13;14;15;16;17;18;19;20;21	friends

Table 2: *A Table showing which person entities are linked from the four articles.Created in Google drive.*

Table 2 is used to reveal the links between entities in the four articles. In addition, this technique of organizing the data, allows an analyst to compute a graph of these entities. The From and To columns contain the id numbers from table 1, and table 2 uses the id numbers as a reference to show the relationship between person entities, and lastly, labels these links as friends, to provide a better meaningful description of the links between the person entities.

Table representing organisations listed in articles

Articles	ID	Entity (Name of organization)	label
1	A1	Cosatu	organization
1	A2	ANC	organization
1	A3	National Treasury	organization
1	A4	SACP	organization
1	A5	NEWAHU	organization
2	A6	Parliament	organization
2	A7	SABC	organization
2	A8	State arms firm	organization
2	A9	Post office	organization
2	A10	Denel	organization
3	A11	IPID	organization
3	A12	Wynberg Regional court	organization
3	A13	Delft police station	organization
4	A14	SABDC	organization
4	A15	SABF	organization
4	A16	Ghana book development council(GBDC)	organization
4	A17	National book fairs in africa	organization
4	A18	APNET	organization
4	A19	Pan African Collaboration	organization
4	A20	Johannesburg review of books	organization
4	A21	Kenyan Literary Magazine *Lolwe*	organization
4	A22	Brittle Paper	organization
4	A23	James Murua Africa Literature Blog	organization
4	A24	SABF	organization
4	A25	LGBTQ+	organization
4	A26	APO Group	organization

Table 3 on page 4: *A table showing the characteristics of the organization entities found in the four articles.Created in Google drive.*

This table is used as it organises the data in a neat format found in the given articles.In addition, this table provides a clear understanding of all organizations presented in the articles. Furthermore, because this table assigns id numbers to these entities, it allows efficient referencing of the entity and associated attributes for computing a graph of these entities later on.

Table representing links between **person with organization**

Article	From (ID number)	To (ID number)	Type of relationship
1	1	A1,A2,A3	associated
1	2	A4	associated
3	3	-	associated
3	4	A13	associated
3	5	-	associated
3	6	-	associated
3	7	-	associated
3	8	-	associated
3	9	A12	associated
3	10	-	associated
3	11	-	associated
4	12	A14,A15	associated
4	13	A16	associated
4	14	A18	associated
4	15	A19	associated
4	16	A19	associated
4	17	A19	associated
4	18	A19	associated
4	19	A19	associated
4	20	A15	associated
4	21	A15	associated
4	22	A15;A25	associated

Table 4 : *A table representing the links between organizations and person entities from the articles provided.Created in Google drive.*

Table 4 is used because it provides a good way of revealing the relationship between organizations and person entities. In addition it also describes the name of the link between these entities, namely associated. Lastly this is a good way to prepare,and give a clear view of what the links in a graph are.

Results

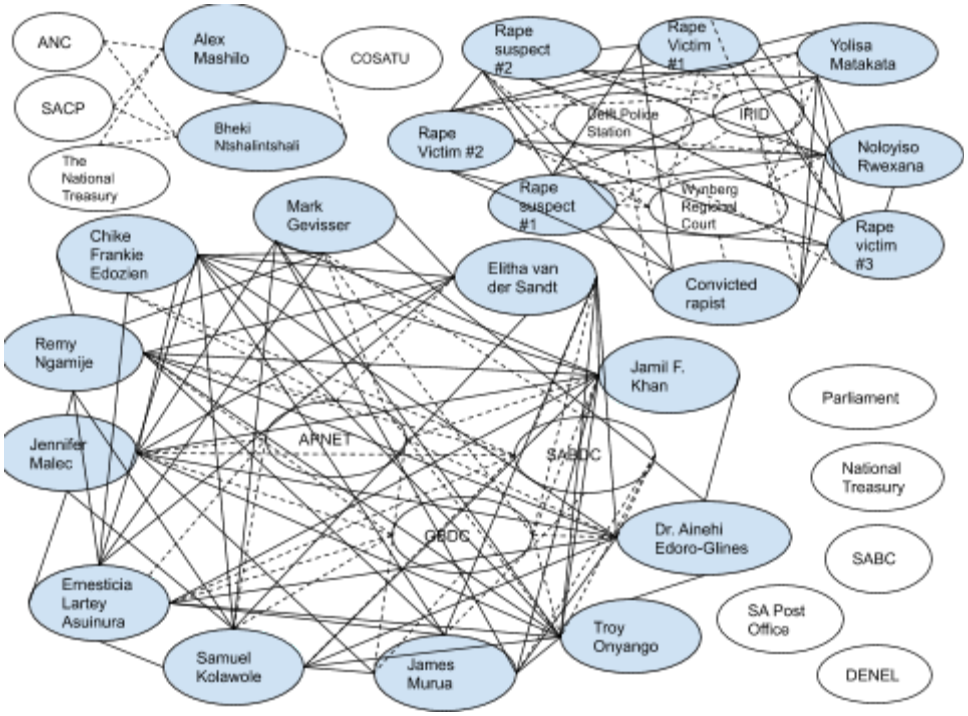
Legend:

- - - - - - - - - - - - : line representing relationship between a person and an organization (association)

——————— : line representing relationship between people (friendship)

: Solid background circle denotes person

: No filled circle denotes organization



The resulting graph obtained is a forest of graphs consisting of 3 strongly connected graphs for articles 1, 3 and 4. It is worth noting that these graphs would all be complete if there were an association between organizations. The graph composed from article 2 is completely disconnected due to it containing no person nodes. The weight of every edge in each graph is

one, due to there being no mention of the same two people or a person & an organization across articles.

Discussion

Based on the results, it is clear that there is no association between the graphs, hence resulting in three disconnected graphs. The four assigned articles resulting in three seperate graphs proves that some stories are connected to each other through the same organizations or people involved. Due to the lack of weighting mentioned above, the significance of each of the relationships are equal in their respective articles and within the graph. From the graph it is seen that, in the future, connections could be made between people and organizations from articles. This is seen to further contribute to the understanding of relationships between news articles and the frequency of these articles, as well as the frequency of the organizations and people mentioned.

Bus services (A knapsack problem)

Methods

In order to solve the bus services problem, it was required to use Java, as well as a dynamic programming approach. Dynamic programming entails breaking up a problem into smaller subproblems, and by storing results into a matrix. The matrix was filled in a bottom-up manner, and the results stored in the matrix are used to find an optimal solution to the problem.

The 0/1 knapsack problem is a perfect example of where dynamic programming could be implemented. In short, the knapsack problem entails: being given an array of values and weights, as well as a maximum capacity, find the optimal solution to maximize one of the fields. An additional calculation would be required in order to determine the optimal solution, in this algorithm the profitability, which is a function of the weight, is used alongside the weight to determine the optimal solution. In this specific instance of the problem, the algorithm would maximize the weight of the various groups, considering that the more weight per bus, the higher the profit.

The algorithm used to solve the problem, was divided into multiple subproblems, namely:

- Creating, and initializing, an object to contain the data concerning the groups.
- Populating multiple arrays for the groups, weights and amount of members with the relevant data, as well as a List-data structure for the multiple groups.
- Iterating through the list in order to populate the matrix with the various different solutions, and then determining the optimal solution, as well as returning it.
- Removing groups from the list, in order to not be reconsidered in iterations to follow, as well as outputting the schools that were both selected and removed.

- Making use of the various iterations to store and output data regarding the results, such as the total weight per bus, the profit per bus, etc.

The group-object was initiated to simplify the operation and flow of the program. The group-objects are later used to populate a list-data structure, this was done to help simplify the iteration process, as well as accessibility of the data stored within. Below is a block of code from the program, showcasing how the object was declared, as well as initialized.

```
// constructor for group objects
public class group {

    private String name;
    private int members;
    private int weight;
    private int profits;

    // constructor for initializing the group object, adding all relevant info, as
    // well as a field for profits to be used later on
    public group(String n, int w, int m) {
        this.name = n;
        this.members = m;
        this.weight = w;
        if (this.members > 100) { // services only charge after the first 100kg, so just a condition to ensure
                                  // there are no non-positive profits
            this.profits = (members - 100) * 5;
        } else {
            this.profits = 0;
        }
    }
    // gets/accessor methods
    public String getName() {return name;}
    public int getMembers() {return members;}
    public int getWeight() {return weight;}
    public int getProfits() {return profits;}
    // basic to string for neater printing, taking into consideration the switching
    // of members and their weights!!!!!
    public String toString() {
        return "Group " + name + ": " + weight + " members, with a weight of " + members+"kg.";
    }

}
```

Image showcasing the group object class

The actual knapsack problem is solved by creating a method that iterates through every possible outcome. As the method iterates, by using nested loops and conditional statements, it populates a 2-dimensional matrix, in a bottom-up manner. A bottom-up approach is when the algorithm starts with the smallest subproblems possible, and works it's way up to the more

complex subproblems by using the results obtained from the smaller subproblems. The maximum capacity of the busses are accounted for by a conditional statement in one of the nested loops, namely:

```
for (int y = 0; y <= cap; y++)
```

By limiting the iterating variable y to be smaller than or equal to the variable cap, which stores the maximum capacity, the algorithm ensures it does not pack the busses with more passengers than they can hold. To follow will be the knapsack algorithm used:

```
// method for determining the optimal way to pack/organize the busses
public int knapsack() {

    this.matrix = new int[Groups.size() + 1][cap + 1];

    for (int x = 0; x <= Groups.size(); x++) { // iterates the amount of times so that each group is covered
        for (int y = 0; y <= cap; y++) { // iterates until the maximum capacity of the busses are reached
            if (x == 0 || y == 0) {
                matrix[x][y] = 0;
            } else if (Groups.get(x - 1).weight <= y) { // condition to determine max
                matrix[x][y] = Math.max(Groups.get(x - 1).members + matrix[x - 1][y - Groups.get(x - 1).weight],
                        matrix[x - 1][y]);
            } else {
                matrix[x][y] = matrix[x - 1][y];
            }
        }
    }

    return matrix[Groups.size()][cap]; // returns the optimal/maximized values for the weighting
}
```

Image illustrating the knapsack algorithm

By using an if-statement inside the nested loops, iterating is made easy. The statement comes down to “if the maximum capacity has not been reached, add the previous results data together with the current field, and if they exceed the previous results, store them in the matrix.”

Because the algorithm has to iterate a total of 5 times (considering there are 5 busses to be filled), a method was made to check which groups were selected, so they could be removed from the list, in order to not have multiple duplicate values/groups on different busses. This method was also used to gather data on the total weight per bus and the total profit made per

bus. To follow is the method used to remove groups, and output useful data regarding the solution:

```
// method used to remove schools from the list after being considered
public void removeGroups() {
    int totalProf=0;            // variables for tracking the total amounts
    int totalWeight = 0;

    for (int x = 1; x <= 5; x++) {      // only iterating 5 times, because there are only 5 busses
        String removedGroups = "";      // variable used to accumulate all the schools being considered per iteration
        int index = 0;                  // variable used to remember the index of schools to be removed
        int removeable[] = new int[Groups.size()];      //  list used to store the groups to be removed

        int finalProf = 0;              // variable used to accumulate all the profits
        int finalWeight = 0;            // variable used to accumulate all the weights

        this.knapsack();                // calls knapsack for each given matrices, remembering that the matrices changes after each iteration

        int w = cap;                    // use this variable in order to be able to iterate

        for (int y = matrix.length - 1; y > 0; y--) {
            int matSingle = this.matrix[y][w];
            if (matSingle != this.matrix[y - 1][w]) {
                finalWeight += Groups.get(y - 1).members;       // remembering that the weight and values are switched for this problem
                removedGroups += Groups.get(y - 1).name + " ";
                finalProf += Groups.get(y - 1).profits;


                w -= Groups.get(y - 1).weight;
                removeable[index++] = y - 1;        // remembers the index of the group to be removed
            }
        }
        totalProf += finalProf;
        totalWeight += finalWeight;

        System.out.println("Bus " + x + " has a weight of: " + finalWeight + "kg and thus a profit of: R"
                + finalProf + "\nThe included groups on bus " + x + " are: " + removedGroups);

        finalWeight = 0;        // clear/reset these variables, so that they are accurate for next iterations
        removedGroups = "";
        finalProf = 0;

        for (int z = 0; z < index; z++) {       // loop to iterate 5 times to remove the schools
            Groups.remove(removeable[z]);
        }
    }
    System.out.println("\nThe total profit made by the 5 busses is :R"+totalProf+", carrying a total of "+totalWeight+"kg.");
}
```

Image showing the method made to obtain data from groups, as well as remove schools from the list-data structure

The method calls the knapsack method in order to fill the matrix with values, which are then used. The matrix is access via the following for-loop:

```
for (int y = matrix.length - 1; y > 0; y--)
```

By letting the index start at the end of the matrix, where the maximum values are stored, then decrementing, the algorithm is trying to optimize the solution.

The following line ensures that the weight of the considered group is deducted from the total capacity of the bus (c) and subsequent iterations do not over-or-underfill the bus:

```
w -= Groups.get(y - 1).weight;
```

The following line caters to adding the considered school to an array of schools to be removed from consideration:

```
removeable[index++] = y - 1;
```

Results

By populating multiple arrays and data structures with the relevant data and iterating through these lists by populating a matrix with various solutions, the algorithm was able to determine and show the optimal solution to the problem and solves the bus services problem as to how to load the busses in the most profitable way possible.

The results display the five busses, with the calculated maximum weight that the algorithm calculated using the dynamic approach method. This was done by iterating through the weights of the groups and choosing the maximum total weight of members that the bus could allow without exceeding the limit of people the bus can hold. The results also display the total profit each bus will make. This is done by the algorithm by calculating the cost of excess weight of each group on the bus, this then gives the profit of the bus. Finally, the results also display which groups were chosen for the particular bus.

```
To follow are the values that were chosen by the knapsack algorithm, maximizing the weight per bus, while keeping in mind the passenger constraint per bus:
Bus 1 has a weight of: 1300kg and thus a profit of: R5000
The included groups on bus 1 are: K J D
Bus 2 has a weight of: 750kg and thus a profit of: R2750
The included groups on bus 2 are: H E
Bus 3 has a weight of: 700kg and thus a profit of: R2000
The included groups on bus 3 are: L I B
Bus 4 has a weight of: 600kg and thus a profit of: R2000
The included groups on bus 4 are: N F
Bus 5 has a weight of: 500kg and thus a profit of: R1500
The included groups on bus 5 are: O C

The total profit made by the 5 busses is :R13250, carrying a total of 3850kg.
```

For bus 1, groups K, J and D were chosen. Each of weights: 450kg, 350kg and 500kg respectively. This made a total number of 90 passengers, a total weight of 1300kg and a profit of R5000

For bus 2, groups H and E were chosen. Each of weights: 400kg and 350kg respectively. This made a total number of 68 passengers, a total weight of 750kg and a profit of R2750

For bus 3, groups L, I and B are chosen. Each of weights: 100kg, 300kg and 300kg respectively. This made a total of 96 passengers, a total weight of 700kg and a profit of R2000

For bus 4, groups N and F are chosen. Each of weights: 300kg and 300kg respectively. This made a total of 84 passengers, a total weight of 600kg and a profit of R2000

For bus 5, groups O and C are chosen. Each of weights: 250kg and 250kg respectively. This made a total of 78 passengers, a total weight of 500kg and a profit of R1500

Discussion

Looking at the results that have been obtained above. It is clear that the program that has been generated is the optimal solution to the bus services problem. The results section above evidently show how to load the buses in the most profitable way possible. With bus 1 being the most profitable. The total weight of the luggage was 1300kgs. The least profit was made by bus 5 with a total weight of 500kgs. It is evident that from the comparisons of the buses, the greatest contribution to profit is the weight of the customer's luggage. In order to ensure that the company continues to make profitable income they would have to promote more luggage intake. Marketing strategies play a huge role in this. Monthly or seasonal specials and discounts have proven to increase customer purchase. To further develop the concerns of the topic, the addition of more computers running the program and improving the efficiency of the algorithm will ensure that the program operates at optimum level. Other possible implications of this program include efficiency, it can be used in other areas of studies. For example train cars are linked in a specific order so that they may be loaded, unloaded, transferred, dropped off, and picked up in the most efficient manner possible.

References

0-1 Knapsack Problem | DP-10. (2020, August 22). GeeksforGeeks. https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/?ref=lbp

Dynamic Programming - an overview | ScienceDirect Topics. (n.d.). ScienceDirect. Retrieved October 22, 2020, from https://www.sciencedirect.com/topics/computer-science/dynamic-programming