Santa report

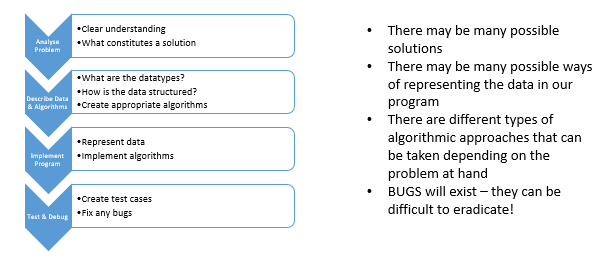
Student name: Galina Lopez

Student number: 10333429

CA4: Helping Santa

Module: Programming for Big Data

We can think of an algorithm as a set of instructions which we must follow in order to complete a task successfully. The problem solving process can be seen bellow:



Based on the kaggle Santa Stolen Sleigh data set, see https://www.kaggle.com/c/santas-stolen-sleigh - construct a python program using the haversine formula covered in the course to group the presents into buckets and develop an algorithm to deliver the presents in as short as distance as possible.

Please include a detailed document outlining the problem set, the data set, your thought process and why you chose to put the presents into the buckets in the manner in your program. Please also expand on any algorithms that you used when calculating the total distance.

Your document should have a summary outlining the answer you arrived at and any further ideas that you would have liked to try to improve upon your solution.

We worked with the ‘gifts.csv’ excel file which contains 100,000 gifts to start.

The file has the following format:

First row contains the gifts headings: GiftId, Latitude, Longitude, Weight.

The remaining 100,000 rows have each present corresponding values for id, lat, lon and weight.

We ran the trip\_mapper.py that takes the ‘gifts.csv’ file and makes a new one we called ‘gifts\_with\_weight\_lat\_long\_clusters.csv’ that has extra columns that groups the gifts into buckets of sleigh trips. For each trip number of close\_by\_500 and close\_by\_1000 you are guaranteed that the ones with the same number are close to each other - within 500 kms and 1000 kms respectively.

* Algorithm:

Start from the North Pole np = (90,0).

Deliver sleight full of presents – equate to a list.

Deliver in most efficient order – shortest path using the haversine function taking into account weight of sleigh – 10kg and max weight of presents allowed per trip – 990kg.

Back to North Pole.

Load next sleight.

Repeat process again until no more presents to deliver.

* Best case scenario

Deliver all presents in one bucket in one trip – 1 trip.

* Worst case scenario

Deliver 1 present per bucket per trip – 100,000 trips.

* Scenario 1

Group gifts close\_by\_500 into buckets – got 533 buckets.

Starting from North Pole deliver to closest gift location in the closest bucket and continue to the next close\_by\_500 taking into account gift weights. Combined weight of sleigh 10kg and gifts 990 kg (maximum weight per trip is 1000kg) should be <= 1000kg. Once you check weight is still allowed you remove gift from bucket and update total distance and total weight of the gifts. When total weight of gifts exceeds 990kg return to the North Pole without removing last checked gift from the bucket, add to total distance the trip from last location to the North Pole, set the total weight to zero and star process all over again till there are no more gifts left to deliver.

Result:

Total distance: 32873361.6696

* Scenario 2

Changed code in my split\_csv\_file method to group gifts close\_by\_1000 into buckets – got 162 buckets.

if giftsList[9] not in my\_buckets:

my\_buckets[giftsList[9]] = []

my\_buckets[giftsList[9]].append(line)

Starting from North Pole deliver to closest gift location in the closest bucket and continue to the next close\_by\_1000 taking into account gift weights. Combined weight of sleigh 10kg and gifts 990 kg (maximum weight per trip is 1000kg) should be <= 1000kg. Once you check weight is still allowed you remove gift from bucket and update total distance and total weight of the gifts. When total weight of gifts exceeds 990kg return to the North Pole without removing last checked gift from the bucket, add to total distance the trip from last location to the North Pole, set the total weight to zero and star process all over again till there are no more gifts left to deliver.

Result:

Total distance: 44022705.6856

Since by tweaking our algorithm and grouping by close\_by\_1000 got a higher total distance than the grouping by close\_by\_500 I will try if I had more time to make new groupings less than 500 instead of more than 500. Will try less radius by 100s ie 400, 300 etc till I get a worst result and start incrementing them by 50s radius and follow this tweaking process till find optimal solution for my algorithm.

I can add to the trip\_mapper.py the possibility to group gifts by close\_by\_400 etc to test these options in the future, by adding extra for loops to the get\_areas method as follows:

for index, gift\_id in enumerate(values['GiftId']): # index 10 in excel file

find\_close\_by(areas, values, index, clusters, 400)

* Algorithm:

Read the ‘gifts\_with\_weight\_lat\_long\_clusters.csv’ file.

Process the ‘gifts\_with\_weight\_lat\_long\_clusters.csv’ file, split it into 533 different csv files, then read in each csv file to process it in the for loop at the bottom and put each bucket in a list of sleigh rides.

Loop through this list of sleigh rides ie sorted buckets taking each sleigh ride ie bucket in turn, first, bucket closest to North Pole followed by next closest and so on.

Started in the North Pole np=(90,0) and calculated the closest location within that first bucket to it.

Took its weight into account in the total weight, added its distance to our total distance and removed it from our bucket.

Took as many gifts as possible as long as not over 990kg as took sleigh weight – 10kg into consideration as well.

Delivered them in one trip planning the route with the shortest path, as I did so I added them to my final route variable for displaying purposes.

Came back to the North Pole each time weight limit – 1000kg was reached, so added this to my total distance and set back my total weight to zero for the next new trip.

Starting in the North Pole again, repeated the process until no gifts were left to deliver.

When the end of a bucket was reached if the weight limit was not reached I took gifts from the next nearest bucket, to complete the trip so as to maximize the load, once again by choosing a gift that will give me the shortest distance from my current location.

At the end printed the total distance.

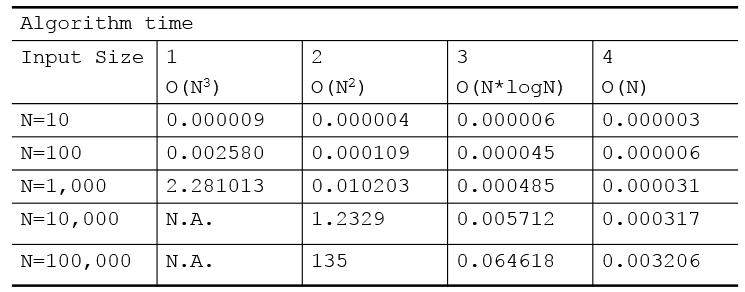
* Big O notation of my algorithm: N^2

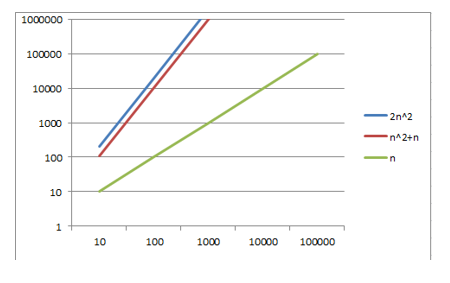
Rather than count the number of operations precisely we can look at the **order of magnitude**.

This way we get an approximation of the time or resources required to solve a problem.

Big O notation is used to express these approximations. In our particular algorithm we have for loop with nested for loop inside, therefore our Big O is N^2, with a typical growth rate: Quadratic.

This is still reasonable in terms of costs for the size of our dataset as the following table shows:





Linear algorithms ie algorithms with Big O = N are best but our algorithm of Big O = N^2 is still acceptable.

If I had more time will make changes to my algorithm so as to make it linear ie Big O of N. By using a mapper and reducer. A mapper to make the buckets which will pipe its results into a reducer to process them ie plan the route and calculate the shortest path.

I developed the unittest suite for my python script and put it in a separate python script file.

I will also find the haversine of the locations and pick ones that are less than 500 km apart and group them like that. Could keep tweaking the number and check how these changes improve or not the final result. By making little adjustments in the right direction we can reach as close as we can the best possible solution for our algorithm.

I added the code to the trip\_mapper.py to add grouping by close\_by\_400 and can change code in algorithm to get this new row in the new csv file ie using index 10 in the giftList to get new total distance result which should give an improvement. No pandas at home to run them and due to lack of time submitting code with grouping by close\_by\_500 instead but made comments in the code to show where changes were made to get optimisations.