**Discrete Mathematics Project.**

**By George Katsonis.**

**Table of Contents:**

* **Introduction**
* **Choosing Destinations**
* **Collecting Data**
* **Historical Background**
* **Definitions**
* **Translation of the Problem**
* **Optimization Algorithms**
* **Applying the Algorithm**
* **Finding the Optimal Solution**
* **Evaluation**
* **Future Trajectories**
* **Conclusion**
* **Appendices**
* **References**

**Introduction:**

The Apollo 11 mission that took place in July 1969, marked a turning point for humanity. For the first time in human history, people everywhere understood the brilliance of human ingenuity, even in the most demanding scenarios **(Loff, 2015)**. Conquering space, apart from being one of the most scientifically important ventures in our times, has proven to be the cornerstone of our modern way of life, with most modern technologies we come to rely upon being the results of our need to overcome the great cosmic barrier **(Garcia, 2018)**. Capitalizing on the positive publicity generated by NASA’s Artemis mission to return humanity to the moon **(*NASA’s Lunar Exploration Program Overview*)** it is the perfect opportunity to launch project Zagreus, the first interplanetary expedition, aiming to establish a transport route between several natural satellites of our solar system. This project, bearing the flag of the European Union, not only will provide invaluable scientific data about the moons visited, but will also be the starting point of the first net for economic growth outside the bounds of earth. After the establishment of the projected outposts in key locations in our solar system, a trading route can be established for any privately owned company to engage in colonizing or mining operations in hundreds of possible celestial bodies, all under EU regulation and taxation. With the rise of privately owned space exploration companies like SpaceX, it is paramount to establish governmental presence in order to avoid an onset monopoly of celestial material usage from any one private organization.

**Choosing Destinations:**

The destinations for this project were chosen based on three criteria, habitability, material excess, and fuel cost mitigation. Saturn’s biggest moon, Titan, is considered one of the most habitable celestial bodies in our solar system, its pressure being accommodating to machinery and its atmosphere providing plentiful material for energy and construction purposes **(Hendrix, 2017).**

From Titan, it is important to also move to Triton and establish an outpost, as it will enable the possibility of quick expansion on the surface, providing a base for launch pads that will enable the exploration of the rest of Saturn’s 82 moons.

It is important to note, that the distance between Saturn and Jupiter, is larger that the distance between Jupiter and Earth. Thus, setting up outposts on Jupiter’s moons is advisable, to maintain a flexible set of destinations within the route. For the same reason it is best to make one more stop halfway from Earth to Jupiter, on a dwarf planet called Ceres.

These stops will allow for cargo exchange, refueling, resupplying, and exchanging information, for all patrons of the trading route, thus establishing a safer, and more efficient economic environment.

Another point worth noting, is that we wish to avoid landing on Jupiter or Saturn. Since they are gas giants, it will be extremely difficult to set up outposts on these planets, while their extreme gravitational fields make launching outlandishly expensive and dangerous. Thus, it is better to make use of the more friendly natural satellites like Titan and Triton orbiting Saturn, as well as Callisto, Ganymede, Europa, and Io orbiting Jupiter.

These destinations, apart from being strategically valuable, are also the most well investigated from previous missions, a fact that will allow us to make use of pre-existing data and technologies, instead of wasting decades performing scouting missions to ensure their viability.

**Collecting Data:**

In this section, the relevant data found in the excel file will be explained as well as all the assumptions and calculations made to reach the presented estimates. Please note that excess data have been included to facilitate further optimizing calculation if need arises. These data will not be explained here, however official explanations can be found [here](https://nssdc.gsfc.nasa.gov/planetary/factsheet/planetfact_notes.html#diam).

The distances between moons and their non-primary planets are calculated as follows: Moon to primary planet, then primary planet to non-primary planet.

For the distances between moons are calculated as minimum estimates, since their proximity along with their short orbits, make it easy to execute trips at maximum efficiency.

For interplanetary trips calculations become inherently more complicated, as they have longer orbits, and many celestial bodies in-between, forcing sub optimal routes to move from one to another. Thus, for these calculations, average distances from the sun are used.

To calculate approximate times for travelling, the speed of the vehicle must be known. Since space vehicles are custom made for every mission, and since this undertaking is to be the first of its kind, all the parameters of the journey need to be established and then the type of vehicle can be approximated.

The first wave of Project Zagreus is to be an uncrewed deep space endeavor. Deep space probes are known to reach extreme velocities when airborne, however our vehicle will not simply pass by the celestial bodies or even orbit them. What we need is something close to a lander space probe like the Mars polar lander **(*1998 Mars Missions*)**.

On the topic of energy, there is a discussion to be made about the composition of the instruments that will power this endeavor. Escaping earth’s gravity is no small feat, as it requires trillions of joules of energy; however, the most challenging task will be to land and take off again multiple times during the trip. To overcome these obstacles, we suggest the following. A delta III type rocket **(*Delta* Britannica)** will be used to carry the vehicle outside Earth’s gravitational influence, then the vehicle will be equipped with ion-engines, to efficiently adjust its course over long periods of time **(*Deep Space 1 In Depth* 2019)**, and a Radioisotope Power System (RPS) to maintain the operation of all electrical subsystems and subsequent launches. Probes could be used to scout the area around each landing zone for materials that can be used to refill the RPS power supply.

All the technologies that will be used in the trip add a significant weight to the vehicle, this fact, along with the limited energy supplies for subsequent launches, means that the average velocity will not be as good as other space probes. A very positive estimate for the speed could thus be at around 50,000km/h.

To calculate the time needed to reach each destination, we will divide the distance with our average velocity.

Finally, for the energy calculations we use the [formula](http://webhome.phy.duke.edu/~rgb/Class/phy51/phy51/node9.html) for escape energy and we assume an approximate weight of 308000 kg from earth since we count the weight of the vehicle and the Delta III rocket, while from other planets we use the approximate weight of 8000 kg.

**Historical Background:**

**Bridges of Königsberg:** The earliest real-world application of graph theory was recorded in 1736, when Leonhardt Euler solved a long-standing practical problem, pondered by the people of Königsberg. The problem asked if a wandering Königsbergian can navigate around town using all the bridges of the town just once and return home at the end. In reaching the conclusion that this venture would be impossible, Euler created the first graph.1

**Around the world:** In 1859, a mathematician called William Rowan Hamilton created a puzzle called “Around the world”, a dodecahedron that labeled its vertices as big cities of the world and tasked the player to find a route through every city exactly once. This process was later named a **Hamiltonian walk** and is largely used even today.

**Landmark problems:** **The traveling postman** is probably one of the most famous problems of graph theory. In this paradigm, a postman aims to find the most efficient route to visit every city just once. This problem is a textbook application of a Hamiltonian walk, and interestingly, so is project Zagreus. Naturally, similar problems exist, that try to make use of a **Eulerian walk**, a process that aims to visit every edge of the graph just once. Both problems make use of **weighted graphs**; however, there is a plethora of problems that focus on the use of **unweighted graphs**. Such problem is the **four-color problem** by **Francis Guthrie** that asks if, using four colors, one can color every nation in a map, so that no neighboring countries are colored the same. It is also important to mention a **Hamiltonian circuit (or closed Hamiltonian walk)**, that performs a Hamiltonian walk returning to the point of origin at the end. Finally, there several chess problems like the **Knight’s tour**, a problem that seeks a route for a Knight to visit every space just once.

**Disciplines:** Apart from the aforementioned mathematical applications, graph theory is used across all scientific fields for purposes of modeling and optimizing data and structures. **In chemistry**, all covalently bonded structures are represented as graphs called constitutional graphs**. In physics**, graph theory is used to model and study the interactions and interconnectivity of the elements in a system. **In engineering and circuit design** graph theory is paramount when it comes to creating efficiently optimized systems in order to minimize manufacturing and operating expenses. Finally, **in computer science**, graph theory is the cornerstone of modern data organization, networking, and machine learning, with memory spaces or hardware representing vertices and data streams or electron flow representing the edges.

**Development and open problems:** As computer science has seen an explosive burst of growth in the latest years following the spread of worldwide networking and artificial intelligence, need for expansion on graph theory is at an all-time high. As interest and resources are being poured into research and development of more effective algorithms and applications of graph theory, it is expected that a large number of open problems are both created and solved on a daily basis, the rapid spread of information having no small influence over this situation.

**Definitions:**

**Graph:** A graph **G** is defined as a collection of vertices **V(G)**, edges **E(G)**, and a relation that associates only two vertices (not necessarily distinct) called endpoints with each edge.  
We illustrate vertices as points and edges as curves connecting their endpoints.

**Loop:** Both endpoints of a curve are equal.

**Multiple edges:** Two or more edgeshave the same sets of endpoints.

**Simple graphs:** Graphs that do not contain loops or multiple edges. In a simple graph we can define an edge as the unordered set of its endpoints **(e = uv or e = vu)**.  
If these conditions are not met the graph is called a **multigraph**.

**Adjacent vertices:** When two vertices are the endpoints of an edge, they are called **neighboring** or **adjacent.**We denote adjacent vertices as **v ↔ u**.

**Finite Graphs:** Both **V(G)** and **E(G)** are finite.

**Weighted Graphs:** All edges are given a weight, meaning a contextual numeric value. The opposite of this would be an **Unweighted graph.**

**Digraph:** All edges are directed. In a digraph uv is not equal to vu.

**Degree:** The degree of a vertex signifies the number of edges to which it is an endpoint.

**Walk:** A sequence of neighboring vertices. In a walk both edges and vertices can be traversed more than once. If the walk ends on its starting vertex it is called **closed**, and it’s called **open** otherwise.

**Trail:** A trail is similar to an open walk; however, no edges can be revisited. This is also called a Eulerian walk.

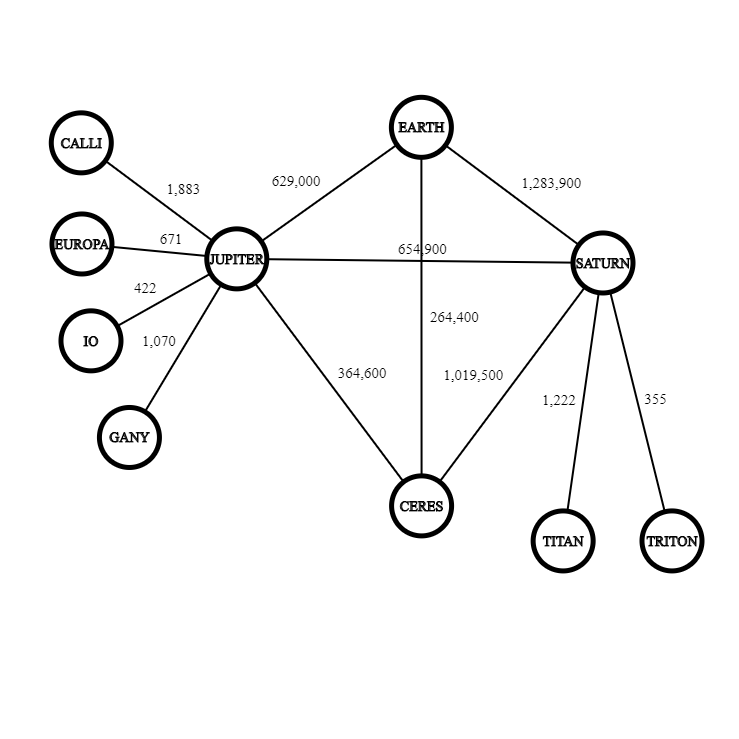
**Path:** A path is similar to an open walk; however, no vertices can be revisited. This is also called a Hamiltonian walk.

**Circuit:** A circuit is a closed trail, meaning we must return to the point of origin in the end.

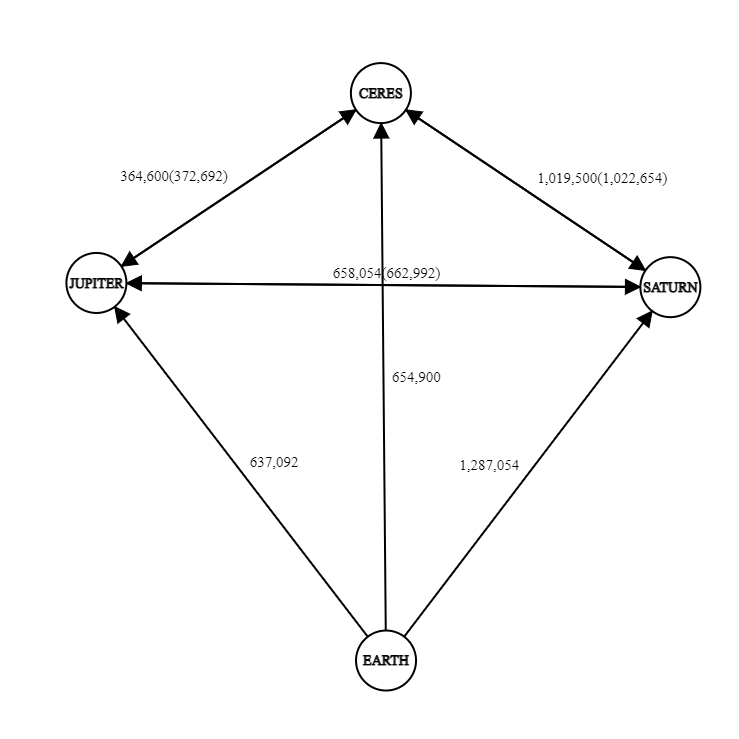
**Cycle:** A cycle is a closed path.

**Translation of The Problem:**

Project Zagreus is a quintessential graph theory problem. By modeling our data in the form of a weighted graph we can apply shortest path optimization algorithms in order to calculate the most efficient route for our expedition, that will have the form of a Hamiltonian walk, not a circuit since for our purposes returning the convoy to earth would be a waste of resources. In the graph, the celestial bodies will be represented as vertices and the path between them as weighted edges, where the weight will be the distance. Since we use planets as waypoints to travel to the moons, we only connect the planets with their moons and the other planets.



However, due to the practical limitations we discussed above, this problem can be further simplified. Since it is in no way viable to not visit every moon of a planet, before moving to the next one, as that would cause back and forth journeys, greatly adding in the total traveled distance, we can assume the following. As soon as the convoy reaches a planted and establishes an outpost on the surface, the primary directive will be to visit every nearby moon before moving on to the next planet. As every trip from moon to moon must be intermediated by a landing on the planet, we concur that the convoy will travel twice the distance to every moon (to and from) before ending up on the planet and moving on to the next. We can then add this distance to the weight of the edges that connect the planet to other planets and end up with this graph.



264,400

It is important to note that the moon distances must only be applied as weights only if the planet is visited for the first time, the weights in the parenthesis are the ones with the added distances. Furthermore, since earth is our starting point and we do not want to return to it, the added distances will always apply. Finally, since we cannot move between Jupiter and Saturn without first visiting one of them, the two possible cases take into consideration the two possibilities of which one was visited first. Thus, our graph now becomes directed with earth as our initial node.

**Optimization Algorithms:**

Finding an optimized route between the vertices of a graph has always been a very important problem, as it allows to model and optimize a very large variety of problems. As such, several algorithms have been developed to try and find this path while being as efficient as possible. The following are some of the most important general use algorithms for this purpose, that apply on directed graphs:

* Dijkstra's algorithm
* Bellman-Ford algorithm
* Nearest neighbor algorithm
* A\* search algorithm
* Floyd-Warshall algorithm
* Brute Force

Undoubtedly, Dijkstra’s algorithm is the most historically significant and widely used algorithm of this list; however, due to the pragmatic simplification of the project and since we seek a Hamiltonian path, the most efficient approach is to use a modified version of the Shortest neighbor algorithm.

The nearest neighbor search (NNS) surfaced during the 70s as the answer to Donald Knuth’s post-office problem, that aimed to assign residences to the nearest post office. As the name suggests, the basic idea of this algorithm is to identify the point of a set, that is the closest to a predefined starting position.

Building upon this basic idea, many iterations of this algorithm have been developed over the years, with a myriad of applications on a multitude of scientific fields, like pattern recognition, DNA sequencing, Plagiarism detection and, most importantly, AI and machine learning. Depending on the variation the complexity of the algorithm changes, but for our implementation we have O(V\*E) complexity where V is the number of vertices and E the number of edges.

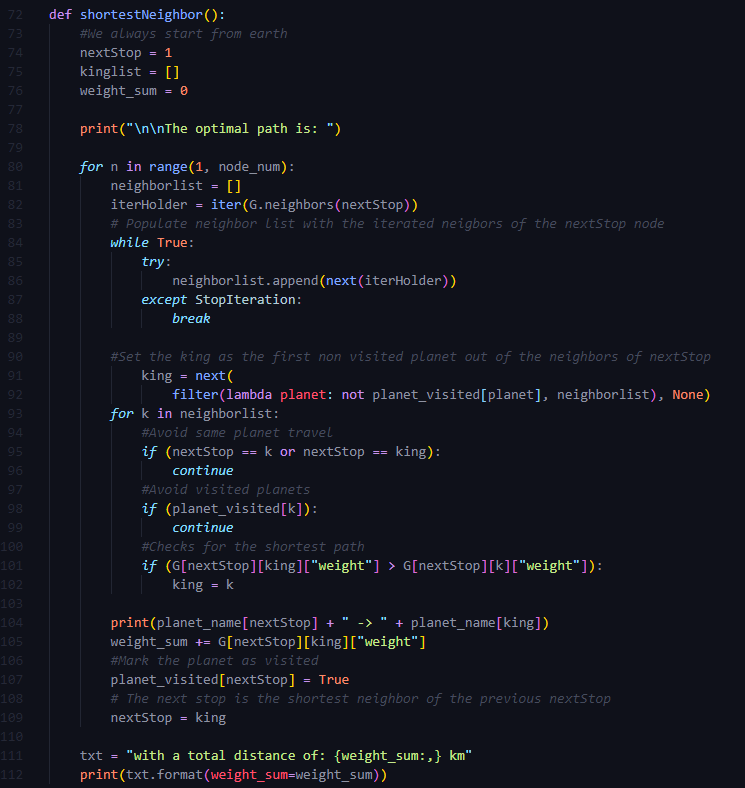
Depending on the application, NNS has adapted to accommodate any number set S of N data in a d-dimensional space; however, such sophisticated tools are beyond the scope and needs of this project. The variation applied here was created from the ground up with only the basic idea of NNS in mind, to be applicable on projects of the same nature, but to also be flexible enough to work with any number of destinations.

It is important to note that NNS is a greedy search algorithm, meaning it favors short term decision making, rather than long term. This means that for more complicated systems, the provided solution might not be the optimal one.

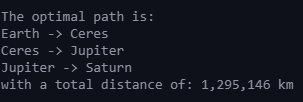
**Applying the Algorithm:**

NNS is applied to calculate the optimal route, when considering distance. In the case that we assume the spacecraft will move with a constant velocity, and we do not calculate the time spend on the surface of each celestial body, calculating the time needed for the expedition would be a simple multiplication; however, in realistic terms the best we could do is very roughly approximate the total time by taking such factors into consideration. Thus, calculating the distance is more appropriate as it stands as an independent measurement that can be used in further calculations. Text version of the code can be found in Appendix 1.

Python Code:



Output:



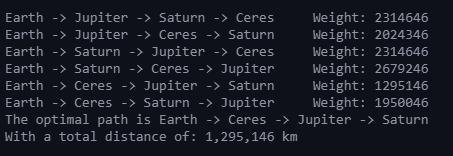
**Finding the Optimal Solution:**

Due to the simplifications of the project (mentioned in part B), the brute force algorithm was very simple to develop and executed very fast. However, since another layer of nesting would be required for each added node, it is clear how fast it would increase the time complexity of the algorithm. Text version of the code can be found in Appendix 2.

Python code:



Output:



As explained in part B, the order in which the moons will be visited after landing on the planet does not matter in terms of total distance. Furthermore, we want to end up on Titan instead of Triton since Titan is highly habitable and an excellent final stop for the trading route. Thus, we will choose the suggested path based on moon radius, as to maximize material gathering between trips to convert to fuel. The suggested path is as follows: Earth 🡪 Ceres 🡪 Jupiter 🡪 Ganymede 🡪 Callisto 🡪 Europa 🡪 Io 🡪 Saturn 🡪 Triton 🡪 Titan.

**Evaluation:**

The NNS algorithm extremely efficient compared to brute force for two major reasons, time complexity and flexibility. As stated above the time complexity for NNS is O(V\*E), while brute force is O(V-1!), making it a lot slower in comparison.

When it comes to flexibility, the brute force algorithm needs to be rewritten for every new node added since we need to compensate with more nested loops, while the NNS can be applied for any number of vertices and edges of any weight.

While the project tries to be as realistic as possible when it comes to the conditions and obstacles in such a travel, it is apparent that an actual endeavor of this magnitude would require hundreds other factors to be considered and included in any planning algorithms. Thus, all the tools in such case would be extremely sophisticated; however, it is very much possible that they would build upon these basic foundations and ideas.

If there were no time constraints, this project would benefit from a GUI application that would visualize the algorithms in real time, the use of more sophisticated tools as a showcase, and a framework that can accept any number of user defined destinations and distances that would then model and solve the problem with those parameters. Finally, from a pragmatic standpoint, factors such as gravitational slingshots and electromagnetic interferences would also be major concerns that would have to be considered.

**Future Trajectories:**

As I am very interested in bioengineering and molecular medicine, one of the most interesting applications for me is the application of graph theory to model and emulate the interactions between proteins in the body, and identify their usage using AI.

In general, the use of AI and neural networks makes extensive use of graph theory since many AI models operate as networks of information.

More specifically an interesting application I drafted a paper on is the application of machine learning to determine the coefficients that lead to the characterization of an age-related protein, by analyzing large databases of age-related proteins and testing for a large variety of proteinic features.

**Conclusion:**

As evident from these approximations, this will most certainly be the greatest scientific endeavor, as well as the greatest engineering challenge, of mankind. Due to the extreme lengths of the journey and the cost the entire operation, it is critical to make proper preparations and proceed with utmost caution.

The lack of energy resources and the fact that such long-distance communications will be difficult, the entire task will need to be manned by sophisticated software to control the vehicle along with the several drones and probes that are going to be included.

Through the application of Nearest neighbor search, we have established an efficient way to calculate the optimal route for our expedition, and we have created the basis upon which the space agency can create the more sophisticated tools for the needed calculations. This will not be an easy task as there are a myriad of factors that need to be taken into consideration before embarking.

However, the great cost, and the long years of preparations will be more than worth the effort, as project Zagreus will mark the definitive first step of humanity toward interplanetary economic growth.

**Appendixes:**

**1.** *#Nearest neighbor method*

def shortestNeighbor():

*#We always start from earth*

    nextStop = 1

    kinglist = []

    weight\_sum = 0

    print("\n\nThe optimal path is: ")

*for* n in range(1, node\_num):

        neighborlist = []

        iterHolder = iter(G.neighbors(nextStop))

*# Populate neighbor list with the iterated neigbors of the nextStop node*

*while* True:

*try*:

                neighborlist.append(next(iterHolder))

*except* StopIteration:

*break*

*#Set the king as the first non visited planet out of the neighbors of nextStop*

            king = next(

                filter(lambda planet: not planet\_visited[planet], neighborlist), None)

*for* k in neighborlist:

*#Avoid same planet travel*

*if* (nextStop == k or nextStop == king):

*continue*

*#Avoid visited planets*

*if* (planet\_visited[k]):

*continue*

*#Checks for the shortest path*

*if* (G[nextStop][king]["weight"] > G[nextStop][k]["weight"]):

                king = k

        print(planet\_name[nextStop] + " -> " + planet\_name[king])

        weight\_sum += G[nextStop][king]["weight"]

*#Mark the planet as visited*

        planet\_visited[nextStop] = True

*# The next stop is the shortest neighbor of the previous nextStop*

        nextStop = king

    txt = "with a total distance of: {weight\_sum:,} km"

    print(txt.format(weight\_sum=weight\_sum))

**2.** def bruteForce():

    minN = 0

    minK = 0

    minL = 0

    king = 0

*#Brute force method.*

*for* n in range(2, node\_num + 1):

*for* k in range(1, node\_num + 1):

*if* (n == k):

*continue*

*if* (k == 1):

*continue*

*for* l in range(1, node\_num + 1):

*if* (n == k or n == l or k == l):

*continue*

*if*(l == 1):

*continue*

*try*:

                    total\_weight = G[1][n]["weight"] + G[n][k]["weight"] + G[k][l]["weight"]

*except*:

*continue*

*if* (total\_weight < king):

                    king = total\_weight

                    minN = n

                    minK = k

                    minL = l

*#Combination Unique to the first iteration.*

*elif* (n == 2 and k == 3 and l == 4):

                    king = total\_weight

                    minN = n

                    minK = k

                    minL = l

                print(planet\_name[1] + " -> " + planet\_name[n]

                + " -> " + planet\_name[k] + " -> " + planet\_name[l] + "     Weight: " + str(total\_weight))

    print("The optimal path is " + planet\_name[1] + " -> " + planet\_name[minN]

        + " -> " + planet\_name[minK] + " -> " + planet\_name[minL])

    txt = "With a total distance of: {king:,} km"

    print(txt.format(king = king))

**References:**

* 1998 Mars Missions. (1998, December). Retrieved January 25, 2021, from https://mars.nasa.gov/internal\_resources/818/
* 60 Years & Counting - The Future. Retrieved from https://www.nasa.gov/specials/60counting/future.html
* Ahuja, R., Mehlhorn, K., Orlin, J. and Tarjan, R., 1990. Faster algorithms for the shortest path problem. *Journal of the ACM*, 37(2), pp.213-223.
* Arya, S., Mount, D., Netanyahu, N., Silverman, R. and Wu, A., 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6), pp.891-923.
* Basics of Space Flight - Solar System Exploration: NASA Science. Retrieved January 25, 2021, from https://solarsystem.nasa.gov/basics/chapter4-1/
* Callisto In Depth. (2020, January 29). Retrieved January 25, 2021, from https://solarsystem.nasa.gov/moons/jupiter-moons/callisto/in-depth/
* Ceres In Depth. (2019, December 19). Retrieved January 25, 2021, from https://solarsystem.nasa.gov/planets/dwarf-planets/ceres/in-depth/
* Csacademy.com. 2021. *CS Academy*. [online] Available at: <https://csacademy.com/app/graph\_editor/> [Accessed 15 February 2021].
* Deep Space 1 In Depth. (2019, July 25). Retrieved January 25, 2021, from https://solarsystem.nasa.gov/missions/deep-space-1/in-depth/
* Delta III. (2020, December 8). Retrieved January 25, 2021, from https://en.wikipedia.org/wiki/Delta\_III
* Delta. Retrieved January 25, 2021, from https://www.britannica.com/technology/Delta-launch-vehicle
* Dunbar, B. (2019, August 19). Europa Clipper's Mission to Jupiter's Icy Moon Confirmed. Retrieved from https://www.nasa.gov/feature/europa-clippers-mission-to-jupiter-s-icy-moon-confirmed
* Encyclopedia Britannica. 2021. *graph theory | Problems & Applications*. [online] Available at: <https://www.britannica.com/topic/graph-theory> [Accessed 15 February 2021].
* Europa In Depth. (2020, October 12). Retrieved January 25, 2021, from https://solarsystem.nasa.gov/moons/jupiter-moons/europa/in-depth/
* Ganymede In Depth. (2019, December 19). Retrieved January 25, 2021, from https://solarsystem.nasa.gov/moons/jupiter-moons/ganymede/in-depth/
* Garcia, M. (2018, July 30). Top Five Technologies Needed for a Spacecraft to Survive Deep Space. Retrieved from https://www.nasa.gov/feature/top-five-technologies-needed-for-a-spacecraft-to-survive-deep-space
* GeeksforGeeks. 2021. *Mathematics | Walks, Trails, Paths, Cycles and Circuits in Graph - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/mathematics-walks-trails-paths-cycles-and-circuits-in-graph/> [Accessed 15 February 2021].
* Hall, L. (2019, July 12). Going to the Moon Was Hard - But the Benefits Were Huge, for All of Us. Retrieved from https://www.nasa.gov/directorates/spacetech/feature/Going\_to\_the\_Moon\_Was\_Hard\_But\_the\_Benefits\_Were\_Huge
* Hendrix, A. (2017, October 16). Confession Of A Planetary Scientist: 'I Do Not Want To Live On Mars'. Retrieved January 25, 2021, from https://www.npr.org/sections/13.7/2017/10/16/555045041/confession-of-a-planetary-scientist-i-do-not-want-to-live-on-mars?t=1611513986525
* Io In Depth. (2020, October 12). Retrieved January 25, 2021, from https://solarsystem.nasa.gov/moons/jupiter-moons/io/in-depth/
* Kerepesi, C., Daróczy, B., Sturm, Á., Vellai, T. and Benczúr, A., 2018. Prediction and characterization of human ageing-related proteins by using machine learning. *Scientific Reports*, 8(1).
* Loff, S. (2015, April 17). Apollo 11 Mission Overview. Retrieved January 25, 2021, from https://www.nasa.gov/mission\_pages/apollo/missions/apollo11.html
* Lyons, V. J., Gonzalez, G. A., & Houts, M. G. (2010). `DRAFT Space Power and Energy Storage Roadmap. *Space Power and Energy Storage Roadmap*.
* Mahapatra, R. and Chakraborty, P., 2015. Comparative Analysis of Nearest Neighbor Query Processing Techniques. *Procedia Computer Science*, 57, pp.1289-1298.
* *NASA’s Lunar Exploration Program Overview*. Retrieved from https://www.nasa.gov/sites/default/files/atoms/files/artemis\_plan-20200921.pdf
* Nuclear Reactors and Radioisotopes for Space. Retrieved from https://www.world-nuclear.org/information-library/non-power-nuclear-applications/transport/nuclear-reactors-for-space.aspx
* Oman, H. (2003). Deep Space Travel Energy Sources . http://doi.org/10.1109/MAES.2003.1183867
* Orbital Elements. Retrieved from http://astronomy.nmsu.edu/nicole/teaching/ASTR505/lectures/lecture08/slide16.html
* Planetary Fact Sheet. Retrieved January 25, 2021, from https://nssdc.gsfc.nasa.gov/planetary/factsheet/
* Prathik, A., Uma, K. and Anuradha, J., 2016. An Overview of application of Graph theory. *International Journal of ChemTech Research*, 9(2), pp.242-248.
* Retrieved from http://large.stanford.edu/courses/2012/ph240/johnson1/
* Retrieved from http://webhome.phy.duke.edu/~rgb/Class/phy51/phy51/node9.html
* Solar System Small Worlds Fact Sheet. Retrieved from https://nssdc.gsfc.nasa.gov/planetary/factsheet/galileanfact\_table.html
* Space and Defense Power Systems. Retrieved from https://www.energy.gov/ne/nuclear-reactor-technologies/space-power-systems
* Space exploration. Retrieved January 25, 2021, from https://www.britannica.com/science/space-exploration#ref237024
* Spaceflight. Retrieved January 25, 2021, from https://www.britannica.com/science/spaceflight#ref219228
* Technology – NASA Radioisotope Power Systems. (2020, December 2). Retrieved from https://rps.nasa.gov/technology/
* Titan - In Depth. (2020, December 9). Retrieved January 25, 2021, from https://solarsystem.nasa.gov/moons/saturn-moons/titan/in-depth/
* Triton In Depth. (2019, December 19). Retrieved January 25, 2021, from https://solarsystem.nasa.gov/moons/neptune-moons/triton/in-depth/
* Trudeau, R., 2015. *Introduction to graph theory*. New York: Dover Publications.
* West, D., 2018. *Introduction to graph theory*. New York, NY: Pearson.