Date: September, 2021

Submitted By – Tenzin Jamtsho 170120101102

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

**ABSTRACT I**

**LIST OF FIGURES II**

**1. INTRODUCTION 1**

**2. PROJECT METHODOLOGY/ MATERIAL & METHODOLOGY 2**

**CONCEPT OF ALGORITHM 3**

**Priority 3**

**Genesis Edge 3**

**CASES ENCOUNTERED 4**

**Case1 4**

**Case2 5**

**FINDING OPTIMAL CYCLE 6**

**4. OBSERVATION & RESULTS 11**

**CURRENT DECISION BASIS AND ITS ISSUES 11**

**PERFORMANCE OF CURRENT DECISION BASIS 12**

**ISSUES WITH CURRENT DECISION BASIS 12**

**5. CONCLUSION AND FUTURE SCOPE 16**

**6. REFERENCES 17**

**ABSTRACT**

The problem was first formulated in 1930 and is one of the most intensively studied problems in optimization. It is used as a [benchmark](https://en.wikipedia.org/wiki/Benchmark_(computing)) for many optimization methods. Even though the problem is computationally difficult, many [heuristics](https://en.wikipedia.org/wiki/Heuristic) and [exact algorithms](https://en.wikipedia.org/wiki/Exact_algorithm) are known, so that some instances with tens of thousands of cities can be solved completely and even problems with millions of cities can be approximated within a small fraction of 1%

Tsp is an open problem .And this project attempts to find a close approximation through use of concepts like priority.the report will cover the issues with the current version algorithm .It will cover the foundation for the algorithm. It tries to take a more deterministic approach to find the approximation in polynomial time.

**LIST OF FIGURES**

2.a genesis edge 3

2.b case 1 4

2.c case 2 a 5

2.d case 2 b 5

2.e test data from google or tools 6

2.d solution to data from google or tools 6&10

2.f solution process to test data from google or tools 7-10

3.a case 1 basis 11

3.b case 2 basis 12

3.c case 2 basis scenario 1 12

3.d case 2 basis scenario 12

3.e current case 2 basis issue 14

3.f current case 2 basis issue resolved 15

**CHAPTER-1**

**INTRODUCTION**

The travelling salesman problem (also called the traveling salesperson problem or TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" It is an [NP-hard](https://en.wikipedia.org/wiki/NP-hardness) problem in [combinatorial optimization](https://en.wikipedia.org/wiki/Combinatorial_optimization), important in [theoretical computer science](https://en.wikipedia.org/wiki/Theoretical_computer_science) and [operations research](https://en.wikipedia.org/wiki/Operations_research).

The initial idea for the algorithm came about with the understanding the if a nodes smallest edge to a node is also the smallest edge to that node from which it is connected to .then it has a high probability of being part of the minimum hamilton cycle .Using this I obtained a very good approximation algorithm with approximation ranging from 1.17 to 1.24

Without drastic change in increase in nodes.

As I was working on this algorithm new concepts like priority of a node adn candidates propped up which inspired me to furter find better approximation leading to the following algorithm which will be mentioned further in the report.

**CHAPTER-2**

**METHODOLOGY**

**Dataset**

The dataset we use is obtained from mathwaterloo.com and <https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.html>

The data is in the form of euclidean distances represented by x- coordinates and y-coordinates in.tsp format .It also comes in 2 dimensional matrix from peeple.sc.fsc.edu.com.

**Data prep**

To make the data usable we need it to be in 2d matrix form .

And extract certain information from.

Here we will introduce the concept of priority and sort the data nodes accordingly.we will then sort the distances of every edge for each node.

# Concept for algorithm

## Priority

We can get the value for how difficult or expensive a node is to get-to by adding the sum of all of its edges of the node.

This give us a ranking of which node is more important to get to at a lower edge weight cause the consequences for not getting to it at a low weight is much higher since than those with low priority since its sum of edges is much lower .

## Genesis edge

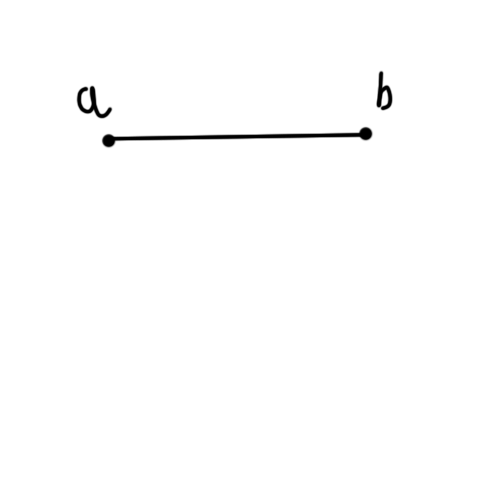
We can almost always guarantee that the smallest edge of the highest priority node or the smallest edge within the network will be a part of the minimum Hamilton cycle. (genesis edge)This will give us the starting 2 points and an edge. We will use the concept 

fig 2.a genesis edge

of two travelers sitting at the 2 starting nodes. Planning out their journeys with a relay in between them to help plan out the shortest way to visit all nodes.

With this we will encounter 2 cases one where the 2 travelers each plan to go to different nodes respectively and dice which one of them should go.or have a clash where they both go to the same node and have do discuss their next course of action

# Cases encountered

### Case1.

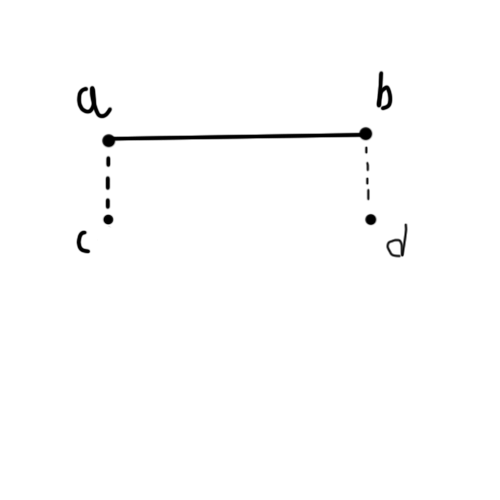


fig 2.b case1

The way to get the nodes **c** and **d** is through a function which returns the node connected to the smallest edge for node **a** and **b** from all remaining nodes. Node b and a are removed from this set to avoid loops since they are already visited. Here the two travellers must discuss which one of them will move to their respective candidates.for node **a**: node **c** /node **b** :node **d**

### Case 2.

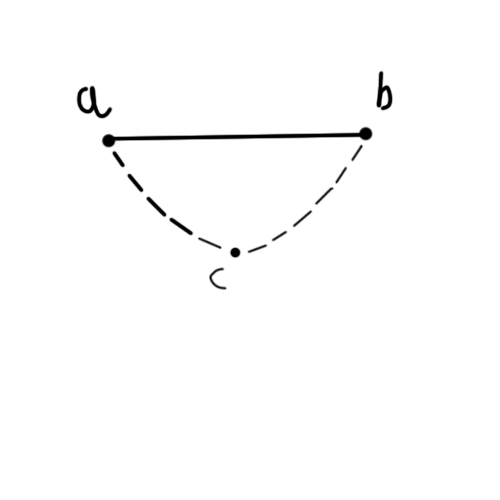


fig 2.c case 2 a

Similar to how node **c** and **d** were obtained in case1 node c if obtained as the next candidate for both node **a** and **b.** Here instead of the two travelers discussing which one will take move to their candidate .they will decide which one of them will remove their plan to journey to common candidate node **c .**For example

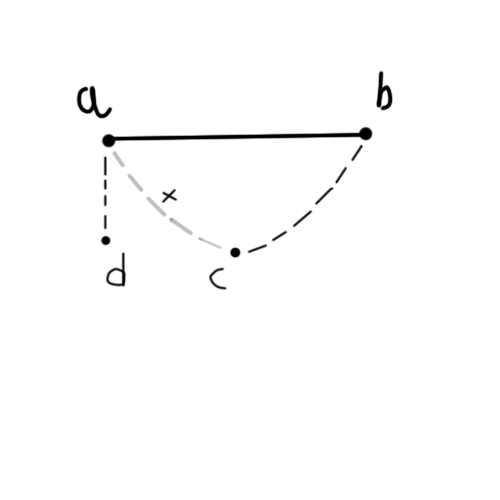
:

fig 2.d case 2 b

if node **a** decides to remove node **c** as a candidate.this will result in **Case 1.**where the two travelers again have to decide which one of them travels to their candidate.

# Finding the optimal cycle

If the correct decision can be made at each turn of either case.1 or case.2 we can get the optimal solution .

For example in the test data from google or tools

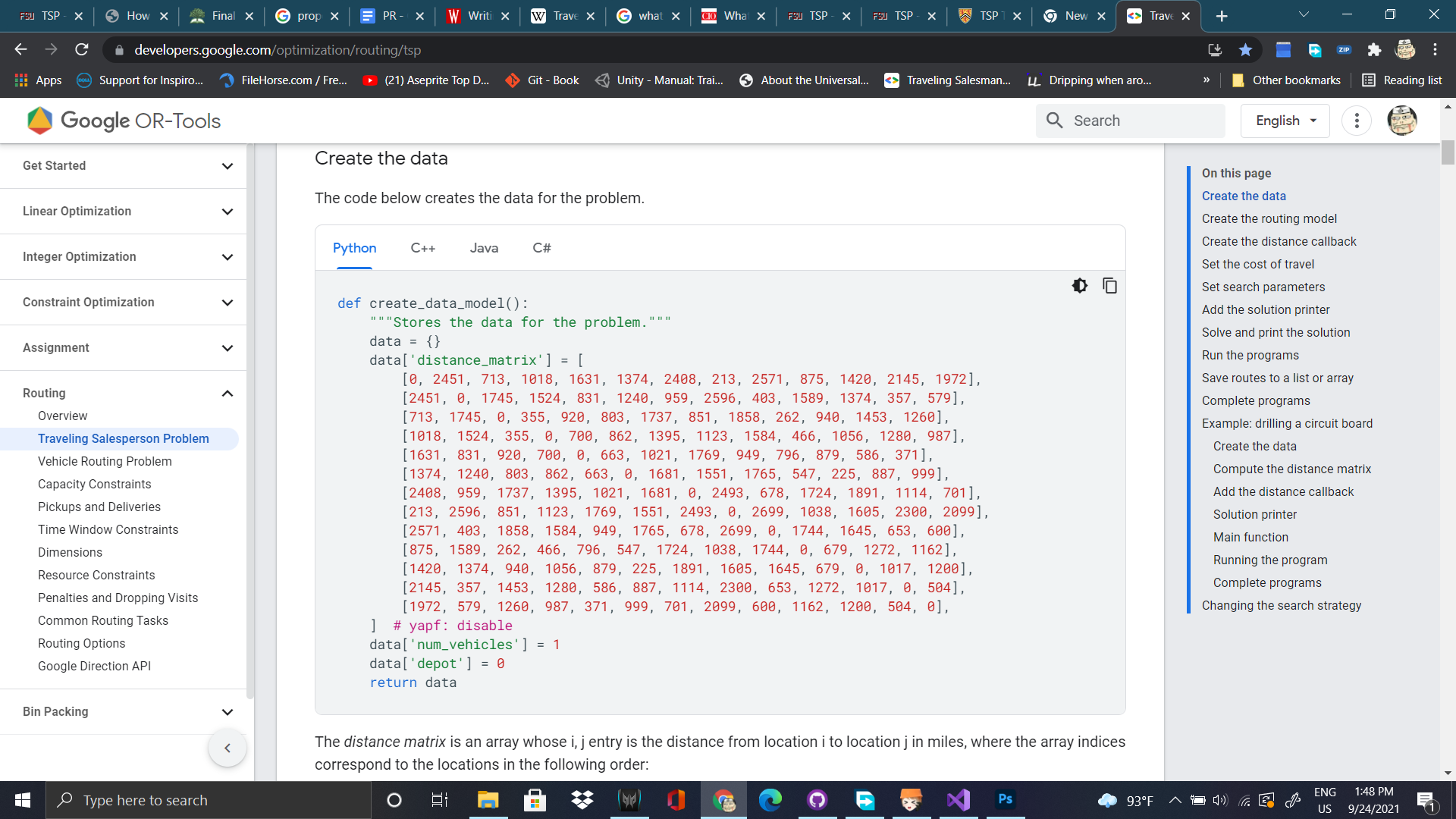


fig 2.e test data from google or tools

With the solution

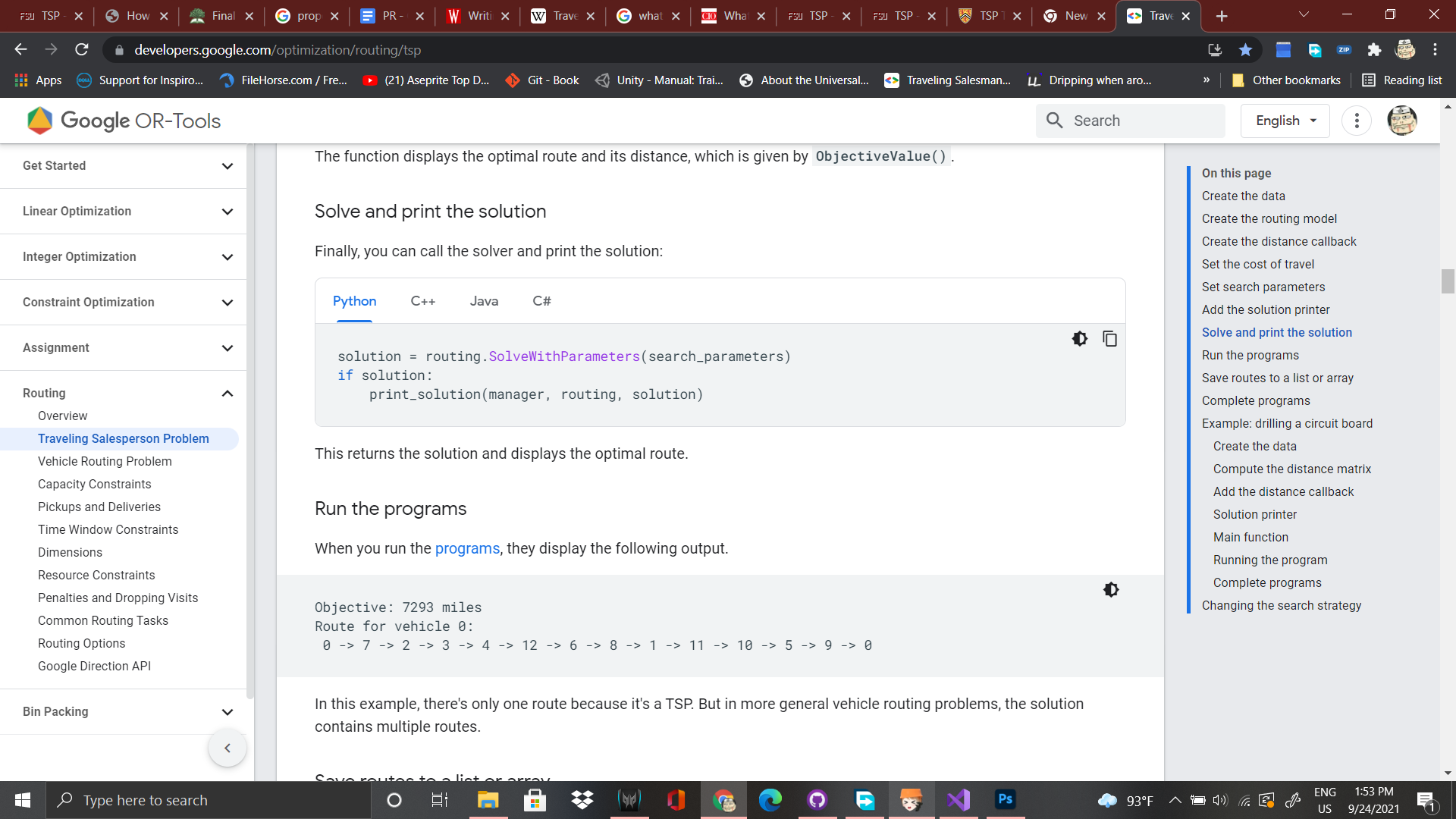


fig 2.d solution to data from google or tools

We get highest priority as node 7 in range of node 0 to 12 with its smallest edge to node 0

**Using the algorithm ignoring the basis for decision for now and supposing we made the correct decision at every point.**

We get the following :

Current solution : 7 - 0

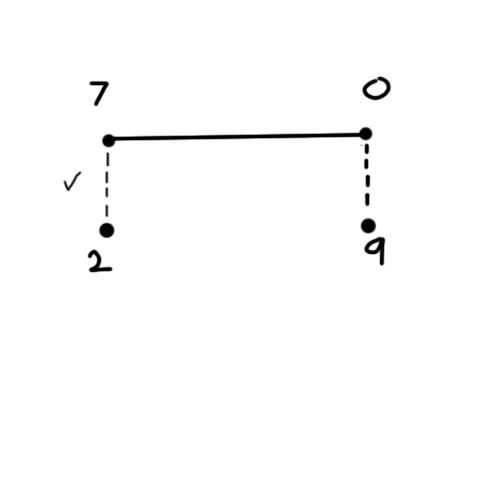
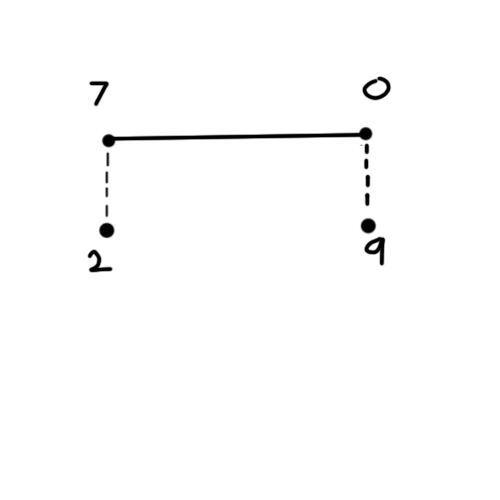
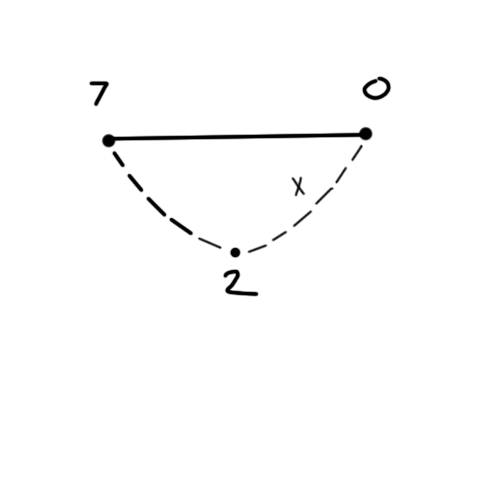
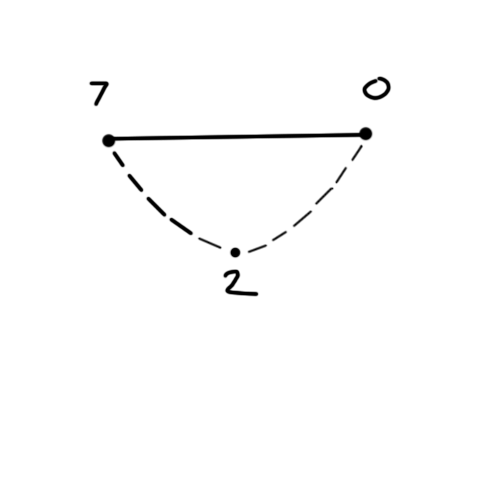


fig 2.f solution process to test data from google or tools

Current solution : 2 - 7 - 0

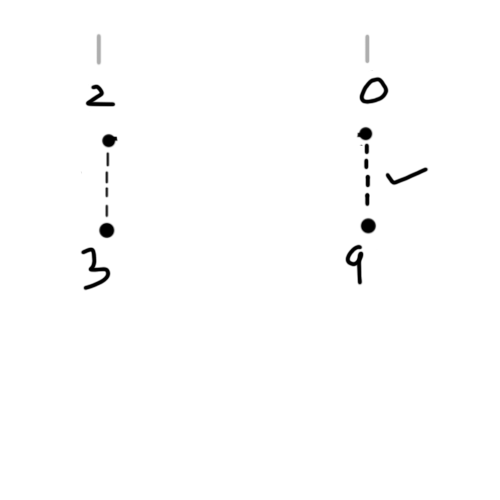
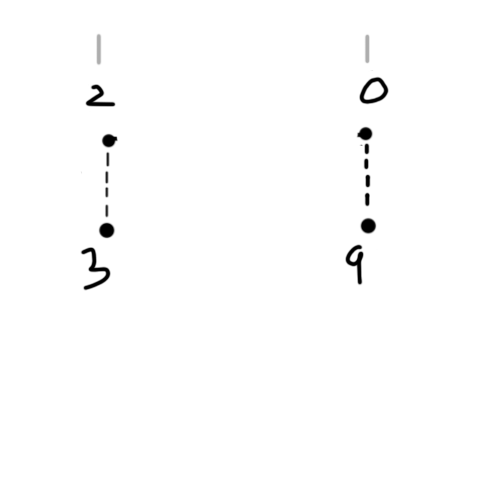
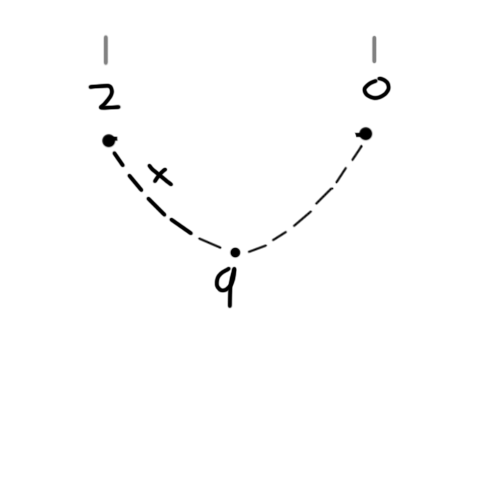
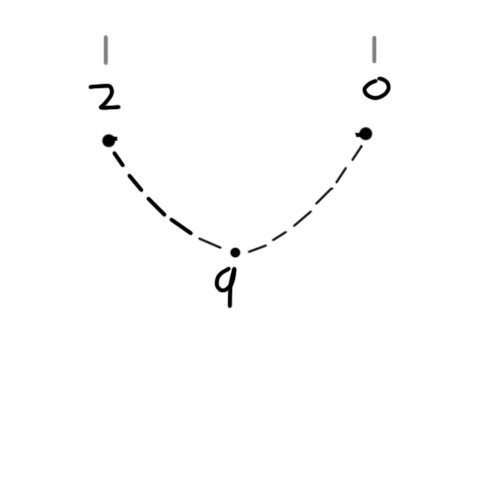


fig 2.g solution process to test data from google or tools

Current solution : 2 - 7 - 0 - 9

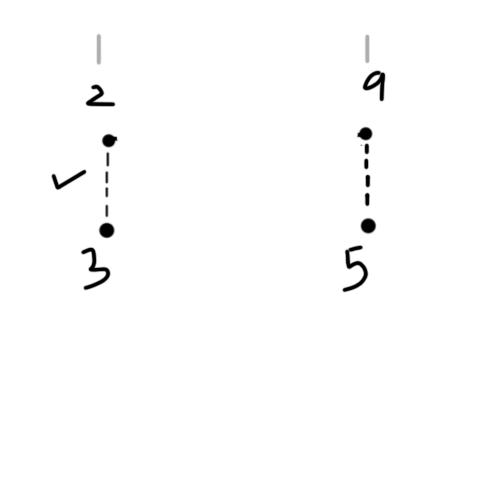
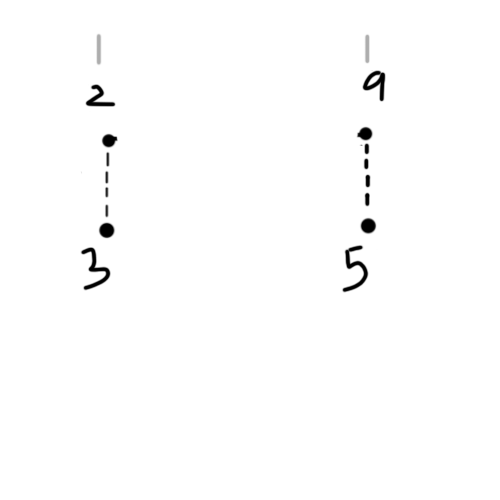
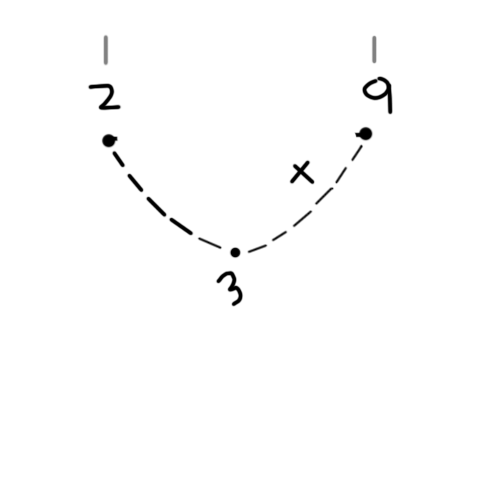
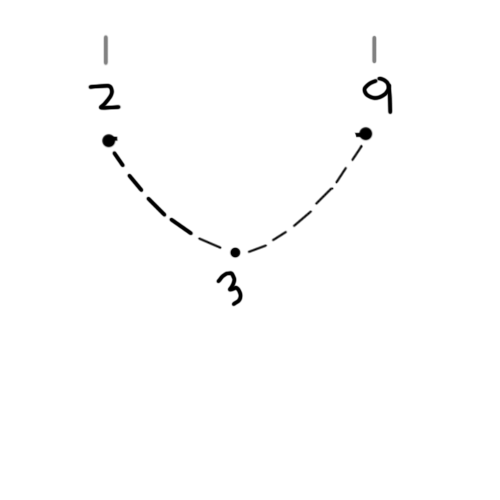


fig 2.h solution process to test data from google or tools

Current solution : 3 - 2 - 7 - 0 - 9

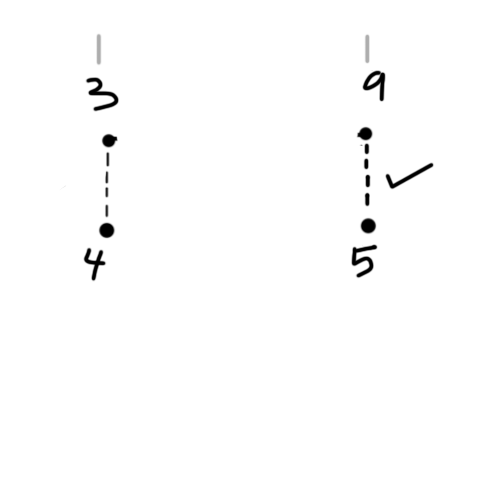
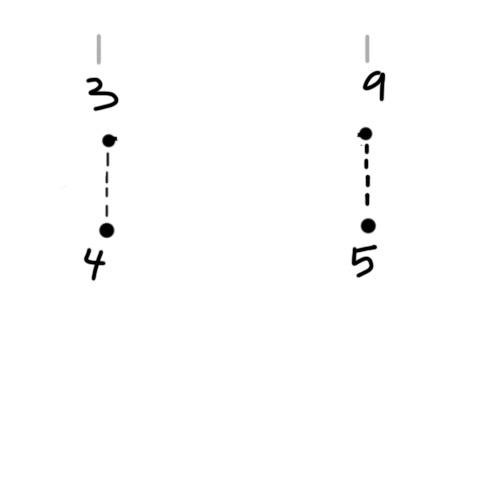


fig 2.i solution process to test data from google or tools

Current solution : 3 - 2 - 7 - 0 - 9 - 5

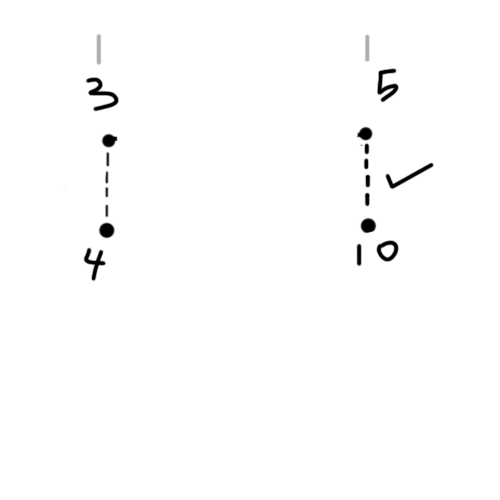
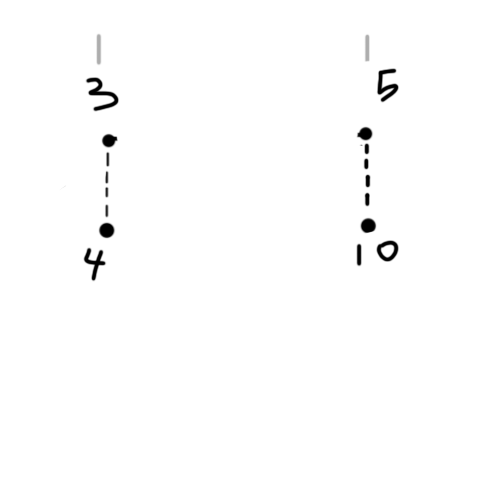


fig 2.j solution process to test data from google or tools

Current solution : 3 - 2 - 7 - 0 - 9 - 5 - 10

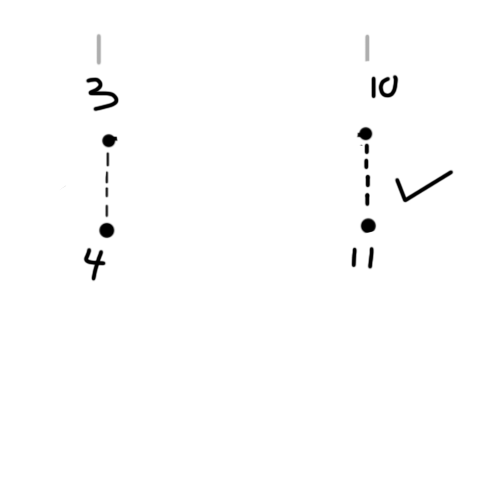
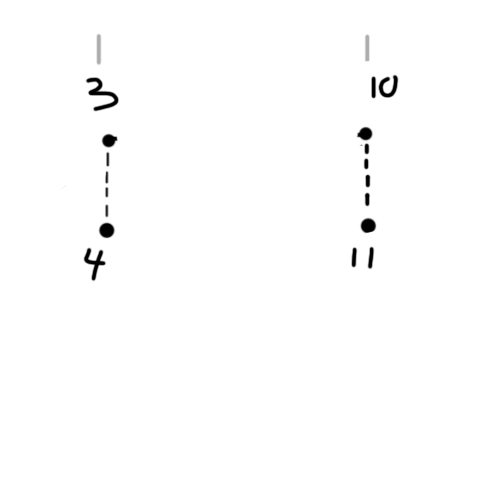
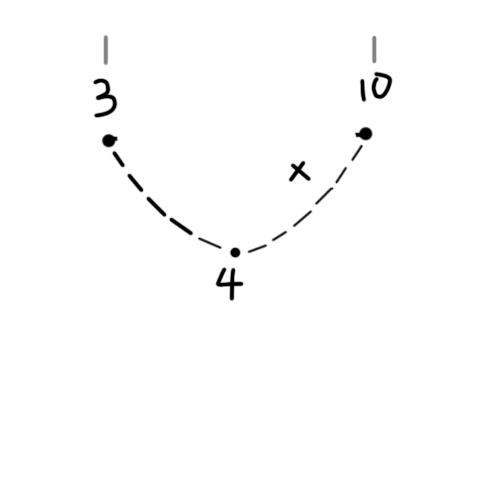
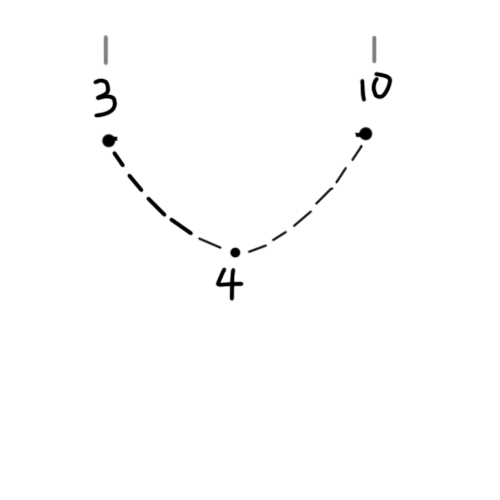


fig 2.k solution process to test data from google or tools

Current solution : 3 - 2 - 7 - 0 - 9 - 5 - 10 -11

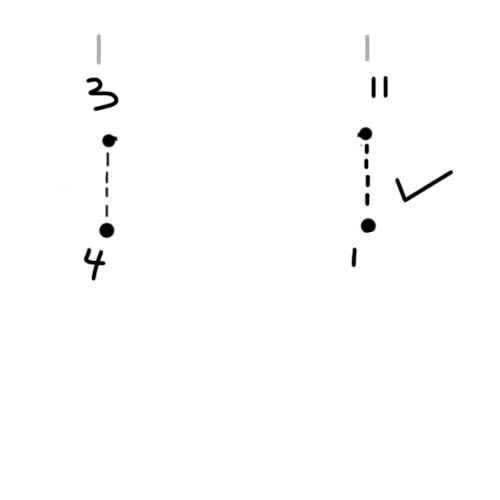
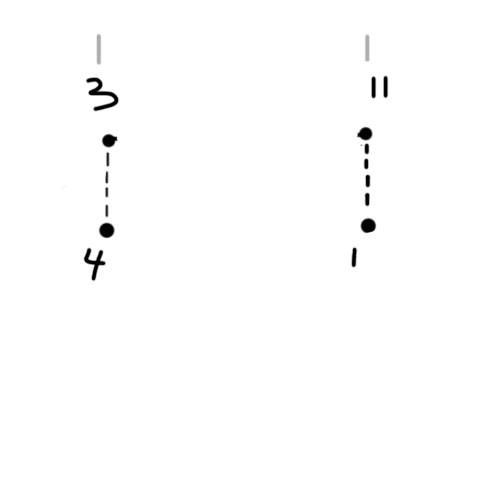
****

fig 2.l solution process to test data from google or tools

Current solution : 3 - 2 - 7 - 0 - 9 - 5 - 10 -11 - 1

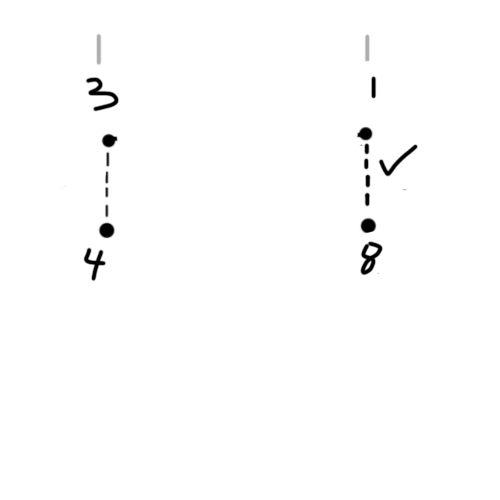
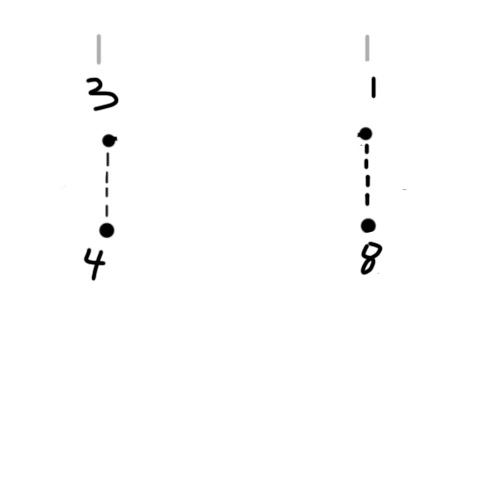
****

fig 2.m solution process to test data from google or tools

Current solution : 3 - 2 - 7 - 0 - 9 - 5 - 10 -11 - 1 - 8

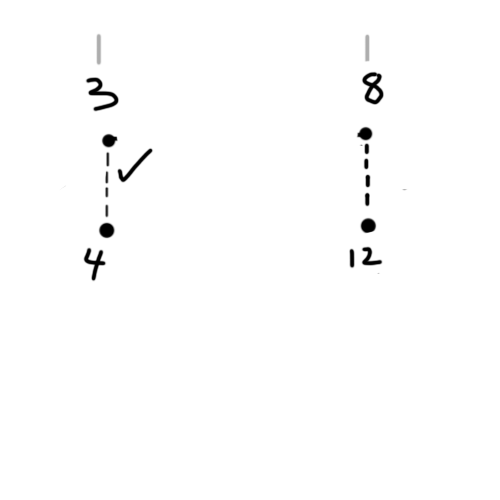
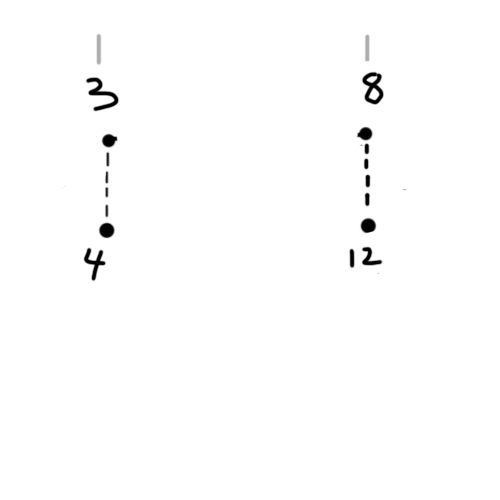
****

fig 2.n solution process to test data from google or tools

Current solution : 4 - 3 - 2 - 7 - 0 - 9 - 5 - 10 -11 - 1 - 8

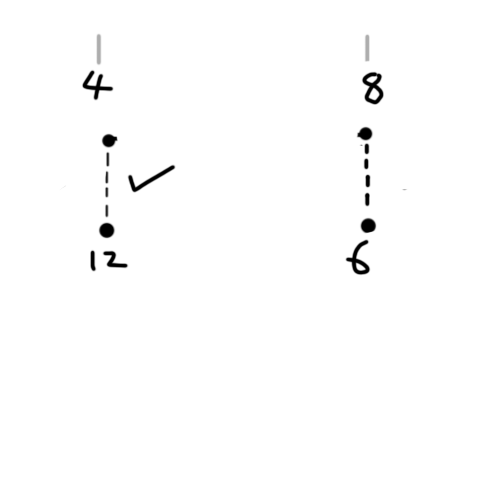
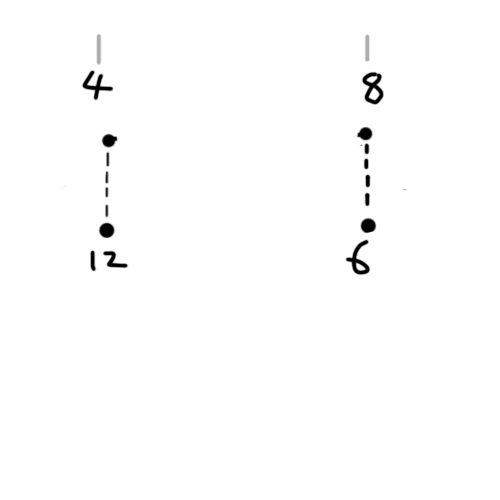
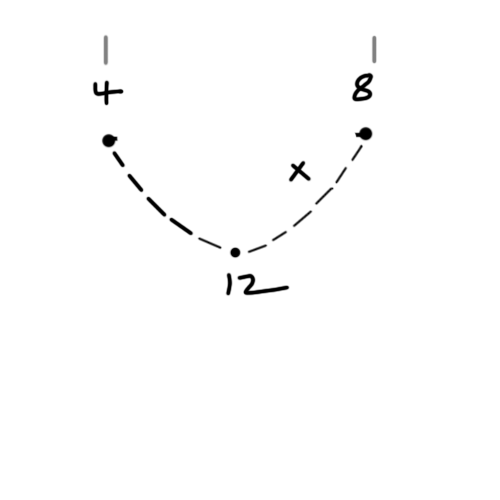
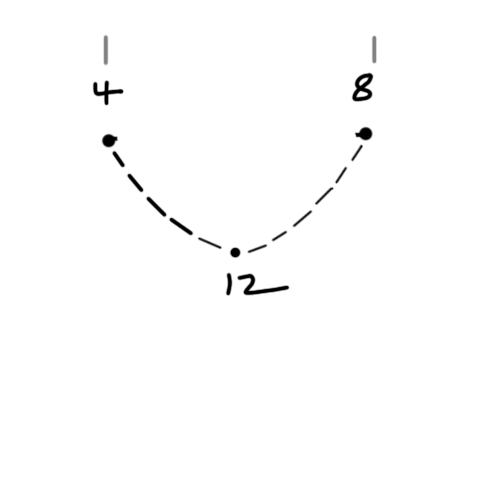
****

fig 2.o solution process to test data from google or tools

Current solution : 12- 4 - 3 - 2 - 7 - 0 - 9 - 5 - 10 -11 - 1 - 8

Leaving only one node so we infer the final solution

Final solution = - 12 - 4 - 3 - 2 - 7 - 0 - 9 - 5 - 10 -11 - 1 - 8 - 6 -

Which is equivalent to the optimal solution.

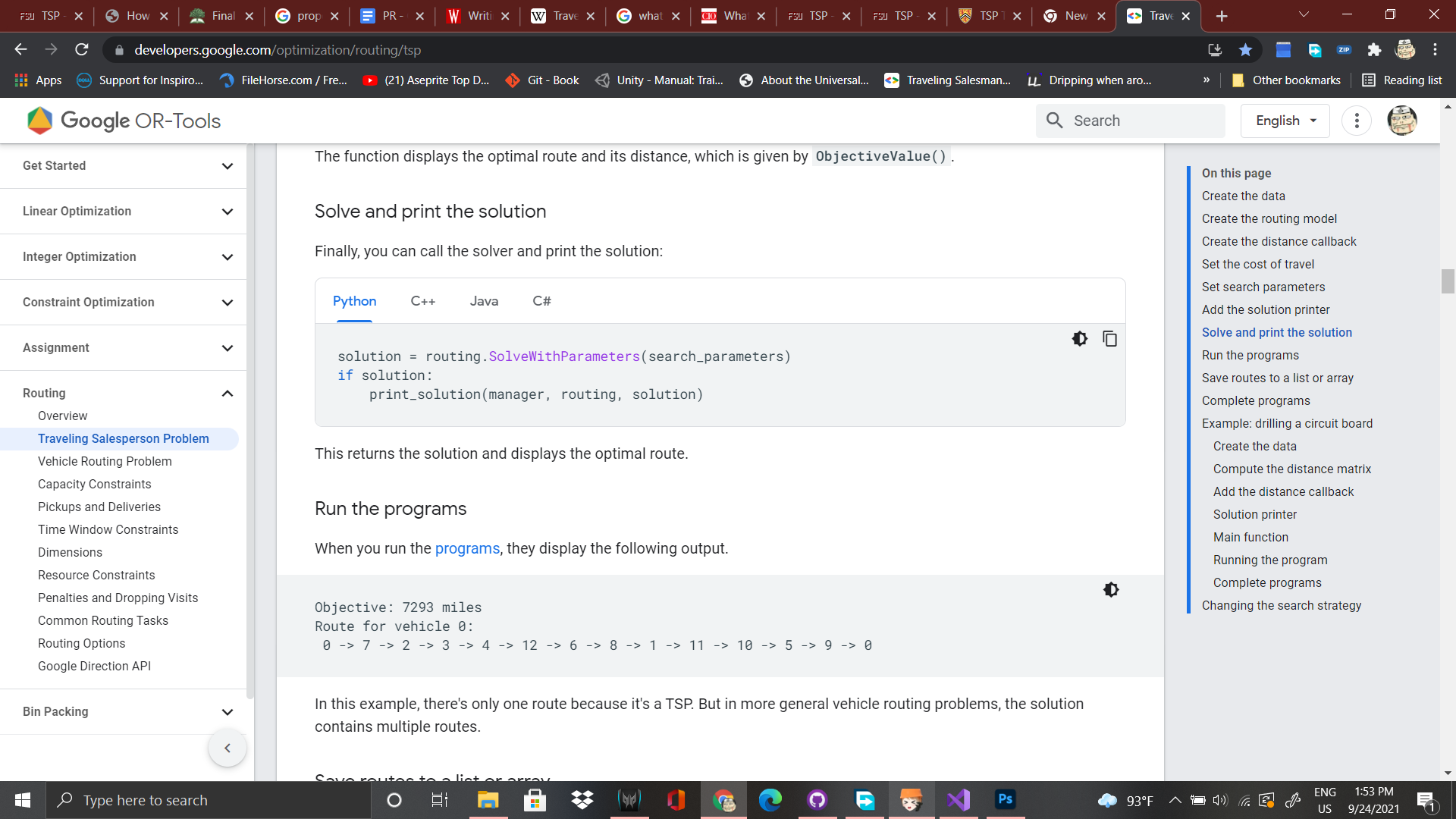


fig 2.d solution to data from google or tools

With this e.g.we prove that if we have the perfect definitive basis for the decision taken at each case we can find the optimal solution.

**CHAPTER-3**

**OBSERVATION AND RESULTS**

# Current basis for decision and its issues:

Initially as I was working on the project the obvious factor for which edge to pick during case 1 and 2 was priority. And the edge weights

Case 1.

In this case the node with the highest priority edge would be picked but this would not always result in the desired outcome as we see later.

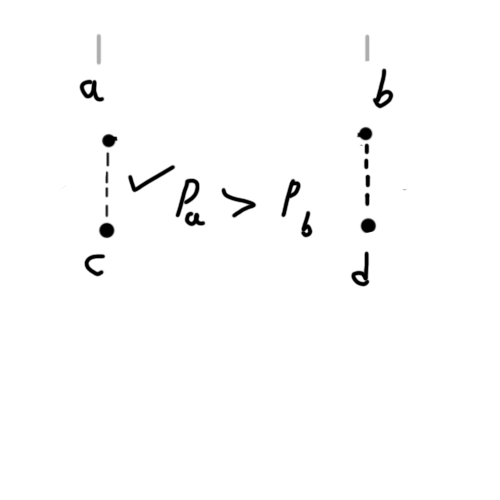


fig 3.a case 1 basis

Case 2.

In this case the future scenarios would be considered and compared.

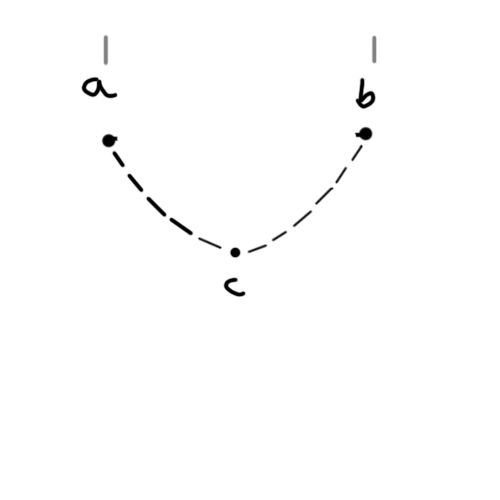


fig 3.b case 2 basis

Scenario 1.

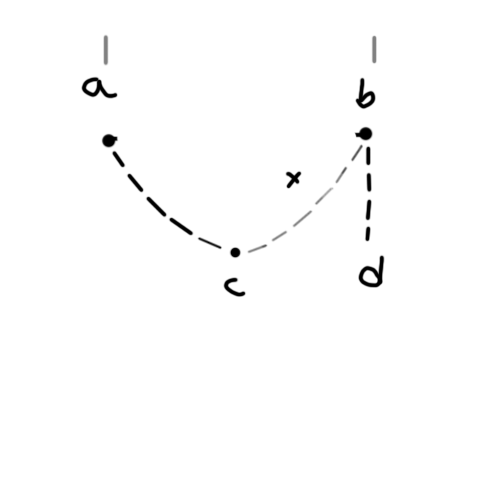
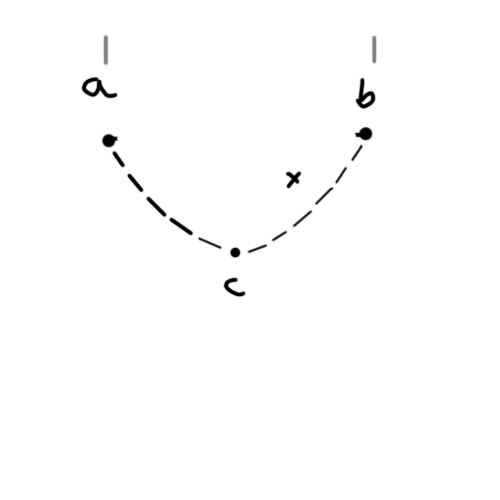


fig 3.c case 2 basis scenario 1

Scenario 2.

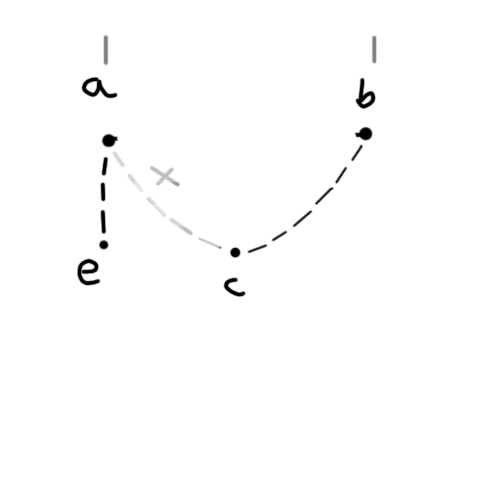
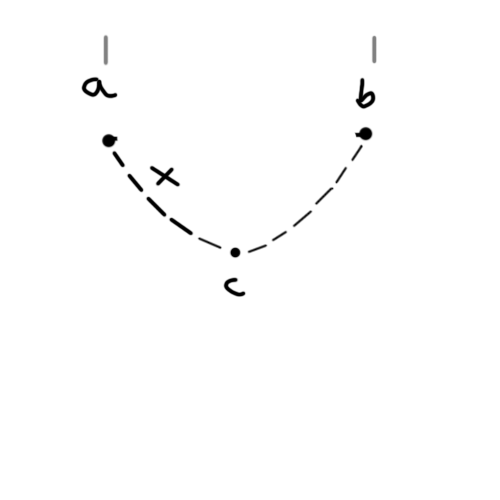


fig 3.d case 2 basis scenario 2

The sum of all edges for each scenario are compared and the one with an edge is picked in contrast to what was mentioned above about discarding the possibility of an edge.Which will be implemented in a future version.

# Performance and issues of current Decision basis:

On the current basis using the smallest edge as the genesis edge on the google or tools test dat we get the optimal solution and on the test set of locational data of djibouti with 38 locations we get an approximate approximation of 1.3 on luxembourg data of 980 nodes .This drastic increase in approximation is due to issues in decision making basis.

tion of 1.002 but a very ba

The future goal of this project would be to decrease this drastic change in approximation value as nodes increase by finding a better decision basis.

# Issues with current decision basis :

## Changing priority not accounted for

It does not take into account that priority should change as new edges are found.the difficulty to reach edges drops caparitiely for different nodes changing which nodes should be prioritised to reach at smaller edges.

## Using only conor node Priority as a factor for Case 1 decision

Though priority is a good factor to decide which edge to pick in Case 1. It doest give us the full picture.Since we only consider the priority of the conor nodes.We may need to consider how picking an edge would affect the priority of the other odes yet to be picked as well.We may also need to consider the value of the edge.but to a lesser degree as we had already sorted all edges for each node thus the edge to the candidates is already smallest possible edge for that node.

## An edge is picked instead of discarding the possibility of the worse edge

We can illustrate this with an example from test data from <https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.html> with 17 nodes

In the solution we have 10 - 9 -1 - 4 and so on If we were to use the current basis 1 to 4 being the genesis edge.we get

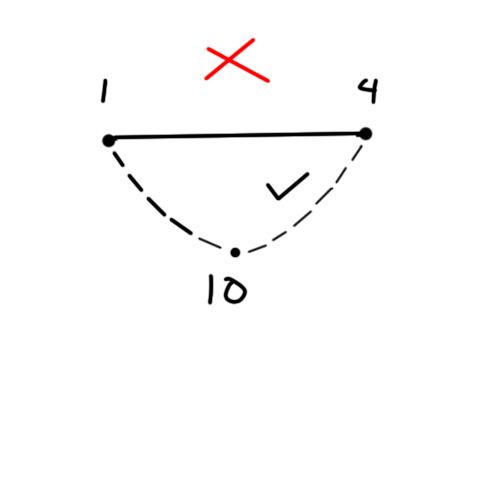
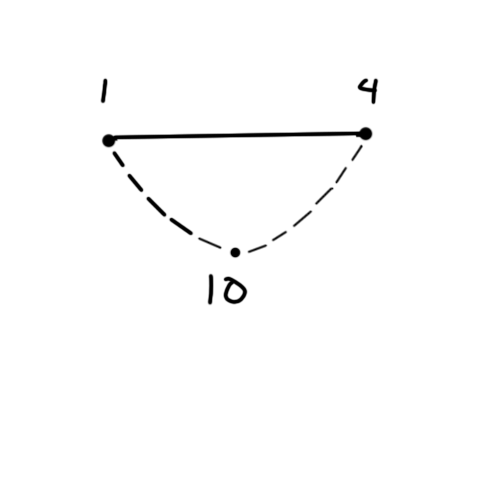


fig 3.e current case 2 basis isuse

At this point we would pick edge 4 to 10 but it is not a part of the optimal solution.

We know both 1 - 4 and 4 - 10 are not optimal.

If instead we had discarded the worse edge we would get the following situation.

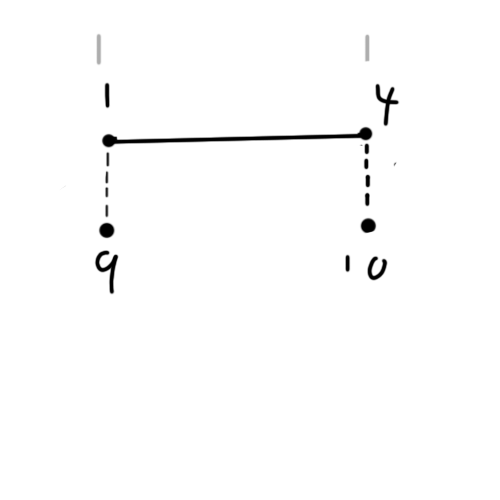
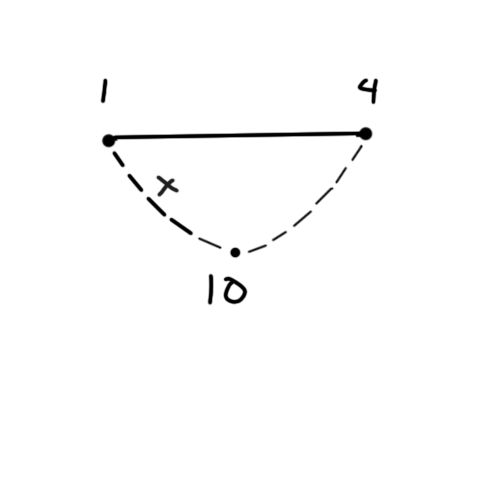


fig 3.e current case 2 basis isuse

Doing it this way gives us much less chance of making an error since discarding an edge still leaves us many possibilities of which edge to pick, instead if we had picked an edge we would be locked in and in case the edge hadn't been part of the minimum cycle the solution would start to derail. Leading to further errors.

Now we can either pick 9 or 10 .If the decision making basis for case 1is perfect the 9 would be picked.

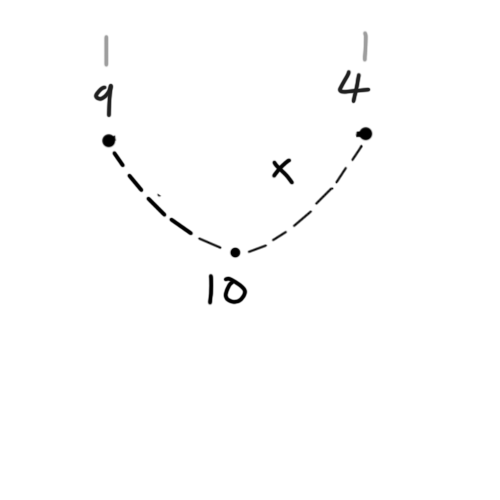
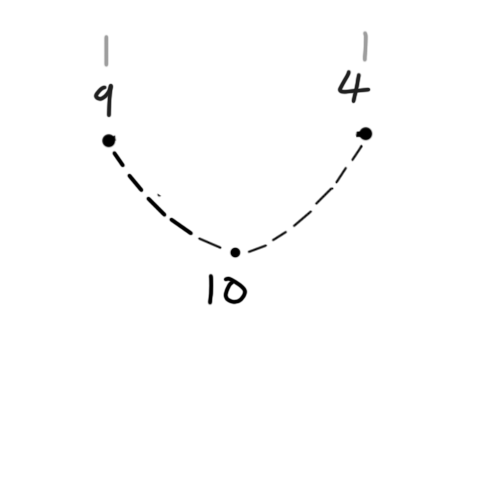
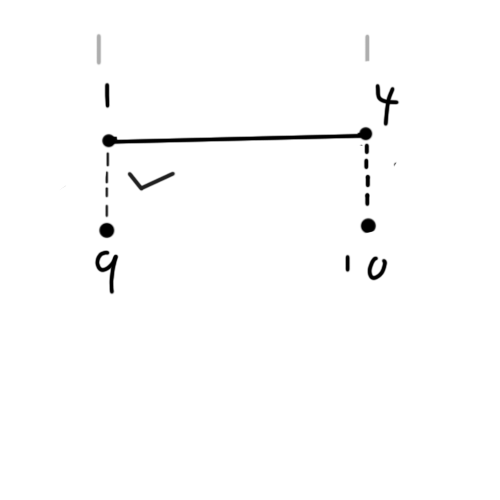
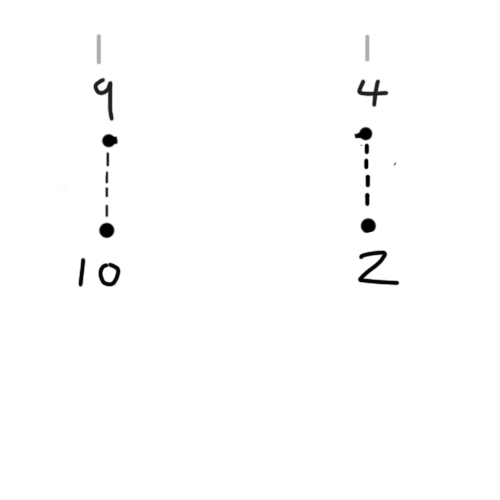
****

fig 3.f current case 2 basis issue resolved

**CHAPTER-6**

**CONCLUSIONS AND FUTURE SCOPE**

In conclusion .If a near perfect decision basis is found then the algorithm, we can almost always get the optimal solution .This too in polynomial time due to the deterministic nature of the algorithm.Even if a perfect basis is not found if we can at least reduce the drastic change in approximation we can gain a really good approximation.

Further research must be done to determine what other factors may affect the decision basis.

**REFERENCES**

1. https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.html
2. https://developers.google.com/optimization/routing/tsp
3. https://github.com/sandwizard/traveling-salesman-console-ver