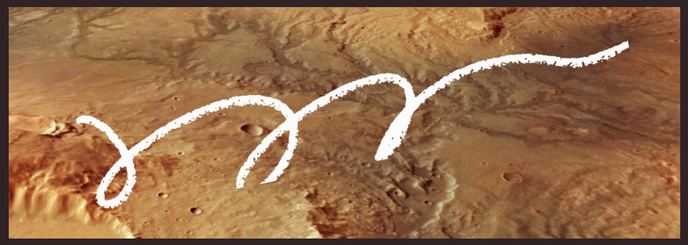
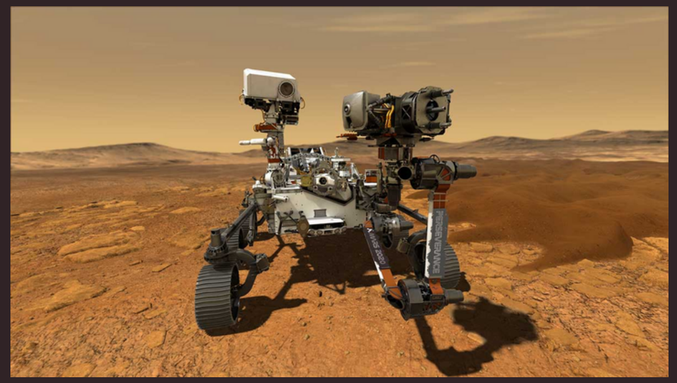


MARTIAN ROVER NAVIGATION SYSTEM



Abhilasha Bansal (2K19/CO/014)
Anshuman Raj Chauhan (2K19/CO/067)

SITUATION

1. PATH PLANNING FOR MARTIAN ROVERS CONSTITUTES OF 2 PHASES - GLOBAL AND LOCAL.

2. LOCAL PHASE IS ACCOMPLISHED WITH THE HELP OF VARIOUS COMPUTER VISION AND DEEP LEARNING TECHNIQUES ON BOARD IN REAL TIME.

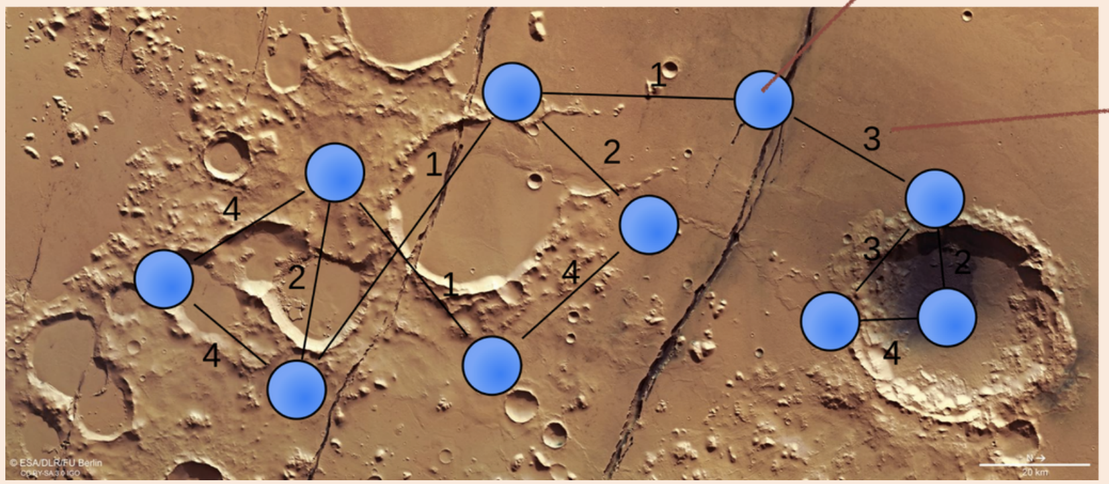
3. GLOBAL PHASE ON THE OTHER HAND HAPPENS THROUGH SATELLITE IMAGES IN ACCORDANCE WITH THE DAILY PATH PLANS DECIDED MANUALLY BY NASA'S JPL.

4. THE DECISION FOR THESE DAILY PLANS CAN BE IMPROVED SIGNIFICANTLY IF A SET OF POINTS WITH ESTIMATED RESEARCH POTENTIALS ARE AVAILABLE BEFORE HAND.

5. THIS PROJECT AIMS AT COMPUTING THE OPTIMAL PATH PLANS PROVIDED RESEARCH POINTS WITH ESTIMATED RESEARCH POTENTIALS, TIME COSTS TO TRAVEL BETWEEN THESE POINTS AND TOTAL TIME BUDGET IS GIVEN AS INPUT.

SIMULATION

Vertices with Research potential.



Edges with time cost as weights.

SYSTEM INPUT

- RESEARCH POINTS AS VERTICES AND THEIR ESTIMATED RESEARCH POTENTIALS AS VERTEX WEIGHTS
- TIME TAKEN TO TRAVEL BETWEEN VERTICES AS EDGE WEIGHTS
- SOURCE VERTEX - DESTINATION VERTEX
- TOTAL TIME BUDGET

SYSTEM OUTPUT

- PATH FROM INITIAL TO FINAL VERTEX THAT FACILITATES MAXIMUM RESEARCH WITHIN THE QUERY TIME BUDGET.
- IF PATH NOT POSSIBLE DUE TO ABSENCE OF EDGES FROM INITIAL TO FINAL VERTEX OR DUE TO LACK OF SUFFICIENT TIME - “NO PATH POSSIBLE” PROMPT.

SOLUTION APPROACH

NONE OF THE STANDARD SHORTEST PATH PROBLEM APPROACH WORK FOR THIS PROBLEM AS THE GRAPH HAS WIGHTS ON VERTICES AS WELL , ALSO A LONGER PATH CAN CERTAINLY YEILD MORE OVERALL RESEARCH.

RECURSION AND BACKTRACKING

- LET'S ASSUME THE SOURCE VERTEX TO BE ‘A’, DESTINATION VERTEX TO BE ‘B’ & TIME CONSTRAINT TO BE ‘T’.
- IN SIMPLEST WORDS, OUR ALGORITHM USES RECURSION ALONG WITH BACKTRACKING TO EXPLORE ALL POSSIBLE PATHS FROM A TO B AND FINDS THE MOST OPTIMUM PATH. WE DEFINE THE MOST OPTIMUM PATH AS ONE THAT FACILITATES MAXIMUM RESEARCH POSSIBLE WITHIN GIVEN BUDGET OF TIME/ FUEL, SOURCE & DESTINATION VERTEX.
- WE DEFINE A RECURSIVE FUNCTION.
- TO FIND THE OPTIMUM PATH FROM ‘A’ TO ‘B’, ‘A’ WILL CALL ITS ADJACENT VERTICES FOR THE PATH WITH MAXIMUM RESEARCH FROM THEM TO ‘B’ AND ONE THAT DOES NOT EXCEED THE TIME CONSTRAINTS.
- THIS MEANS, FOR EACH NEIGHBOURING VERTEX IN THE ADJACENCY LIST OF ‘A’ THAT IS NOT ALREADY PRESENT IN THE CURRENT PATH, WE WILL RECURSIVELY APPLY THIS ALGORITHM TO FIND AN OPTIMAL PATH FROM THAT VERTEX TO THE DESTINATION VERTEX, AND BUILD OUR ANSWER AS THE BEST OF ALL POSSIBLE PATHS THROUGH THESE VERTICES. THE BASE CASE WOULD BE WHEN WE REACH THE DESTINATION VERTEX ITSELF.
- THE PATH WITH MAXIMUM RESEARCH AMONG ALL PATHS WOULD BE CHOSEN AND RETURNED AS THE OUTPUT. THIS IS SIMILAR TO A STANDARD DEPTH FIRST SEARCH IN A GRAPH, BUT HAS MORE CONDITIONS & CHECKS TO ENSURE SELECTION OF OPTIMAL PATH.
- A BOOLEAN ARRAY/ BIT-MASK CAN BE USED TO ENSURE THAT NONE OF THE VERTICES GET REPEATED IN THE PROCESS AND A STATE VARIABLE ‘SPARETIME’ HAS TO BE PASSED IN EVERY SUCCESSIVE CALL TO BACKTRACK AS SOON AS REMAINING TIME BECOMES LESS THAN 0 (IE, TIME CONSTRAINT GETS VIOLATED). THIS WAY WE CAN REDUCE SOME OF THE FUNCTION CALLS(PRUNING THE STATE SPACE TREE) AND HENCE EFFECTIVE RUNNING TIME BY IDENTIFYING PATHS THAT ARE NOT FEASIBLE UNDER GIVEN CONDITIONS.

ALGO Efficiency

Let minimum allowed edge wt. be "min_edge"
maximum allowed time budget be "max_budget"

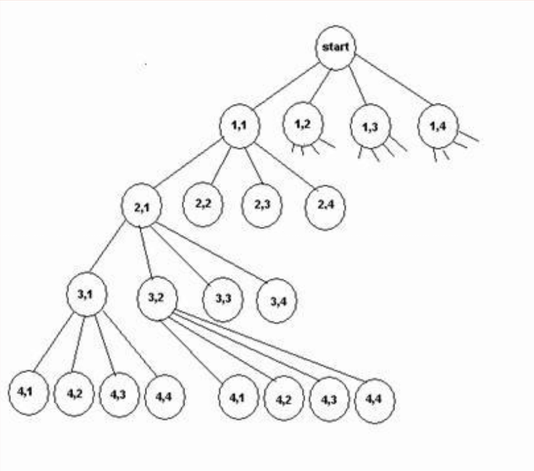
→ Maximum possible nodes that can be traversed without reaching the non-promising nodes of state space tree

$$max\_nodes = \frac{max\_budget}{min\_edge}$$

Let total vertices in graph be "n"

$$T(n) = O\left({}^{n}C_{max\_nodes} \times (max\_nodes!)\right)$$

DESPITE OF PRUNING THE STATE SPACE TREE SIGNIFICANTLY WITH BACKTRACKING THE ALGO WONT BE FEASIBLE FOR MOST CASES.



DP OPTIMIZATION

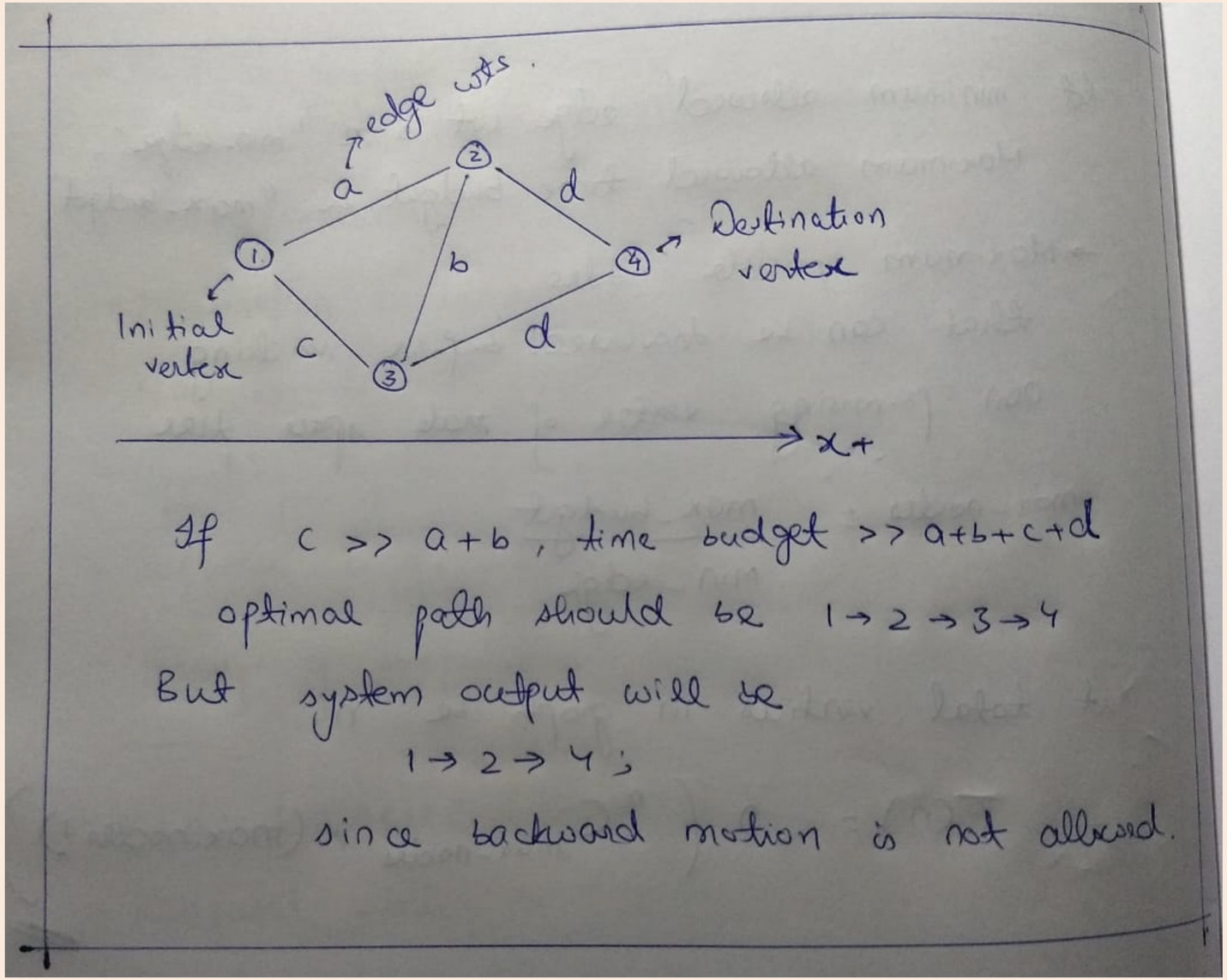
- DP cannot be applied efficiently in the present system as each intermediate vertex from the initial vertex can be reached through a large number of paths; therefore memorisation of any kind will lead to exponential space usage.
- Analysing curiosity's lifetime journey, we observed that the rovers tend to move in the forward direction avoiding any backward movement, thus we constrained our system to allow only forward movement.
- Further we defined
  dp[v][time_budget] as the maximum research that can be performed from vertex 'v' to the destination vertex in 'time_budget'.
  dp[v][time_budget] = max(dp[neigh_v][time_budget - edge_wt(v,neigh_v)]) over all neighbouring vertices 'neigh_v' with greater x-coordinate than 'v'.
- dp[source_vertex][time_budget] will be the overall problem solution.

Curiosity's lifetime journey

ALGO Efficiency

- Time Complexity, T(n) = O((n^2)*time_budget)
- Space complexity, S(n) = O(n*time_budget)
- To ensure a robust system, we have restricted maximum allowed vertices to 10^3 and maximum allowed time budget to 100 units.
- Since in the Martian environment, graphs wont be very large considering the slow motion speed of rovers, this algorithm will be both implementable and effective.
- In case the graph size exceeds 1000 vertices, this algorithm can still be implemented on its subgraphs by dividing the source and destination vertex with multiple checkpoint vertices.

System Drawback



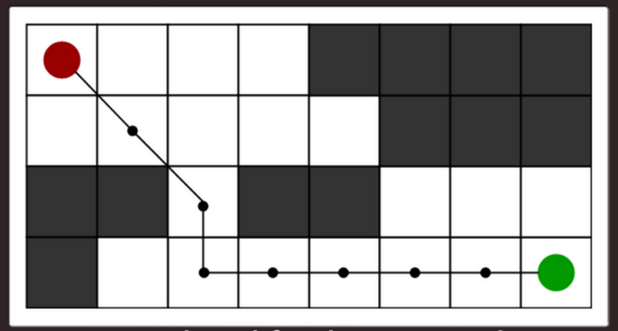
The algorithm wont yield the best result in the demonstrated situation.

But this situation will be highly unlikely as the edge weights between 2 vertices for present rovers are computed using strong deep learning techniques; they will always yield (a+b) as the edge weight between 1 and 3 instead of 'c'.

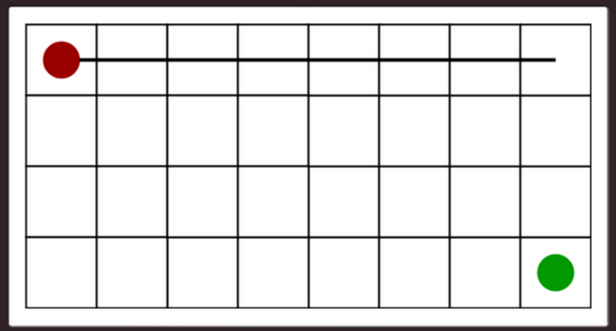
LOCAL TRAVERSAL

Although Local Path planning is done through various Computer Vision and Deep Learning techniques, we have simulated it using A Star algorithm.

- A star algorithm is very similar to Dijkstra's algorithm, but with a heuristics element added.
- Heuristics for points can be generated using : Manhattan distance, Diagonal distance or Euclidian distance.
- In this project the diagonal distance variant has been implemented and tested.



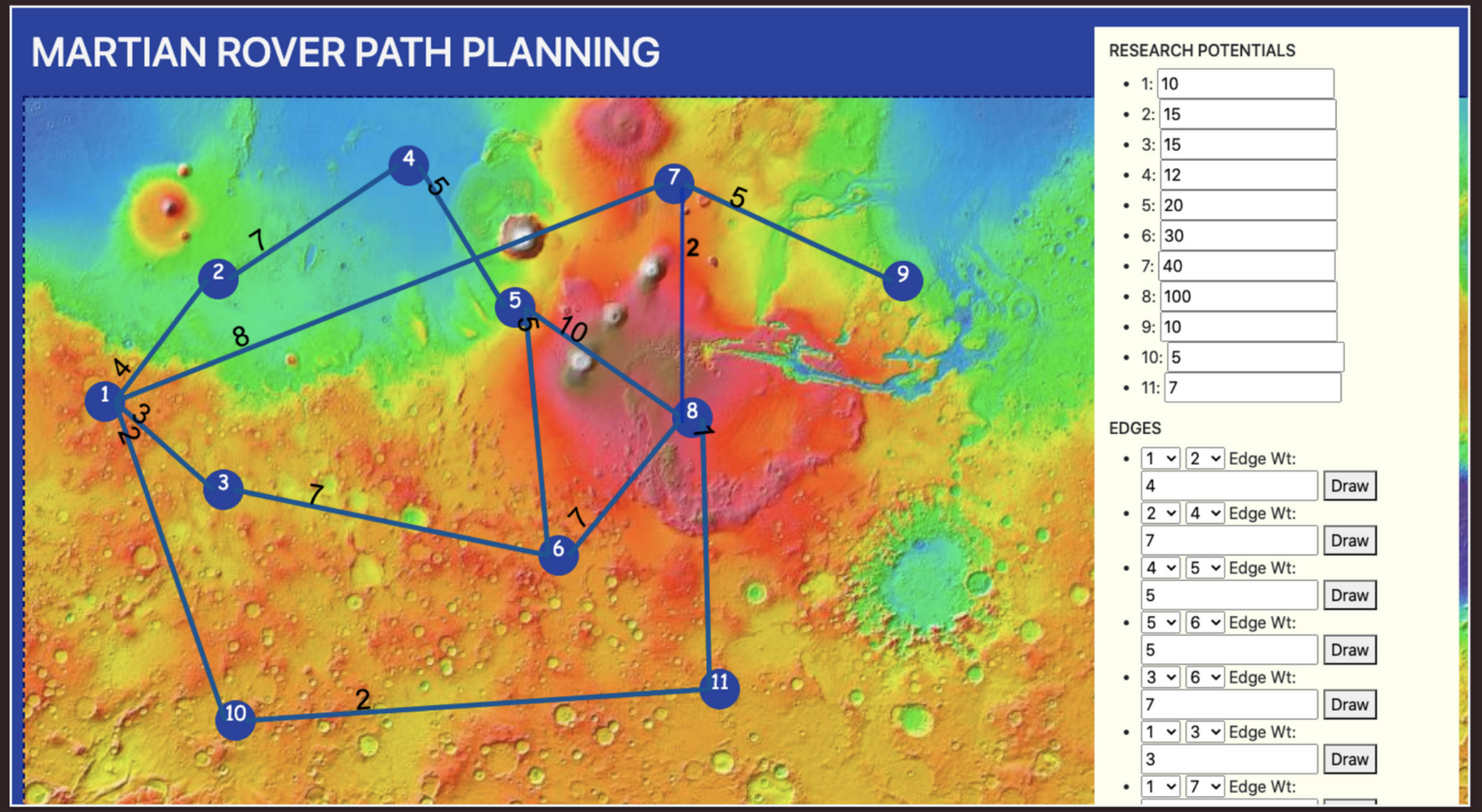
Initial and final points with some obstacles in way



Diagonal distance

SIMULATION AND TESTING

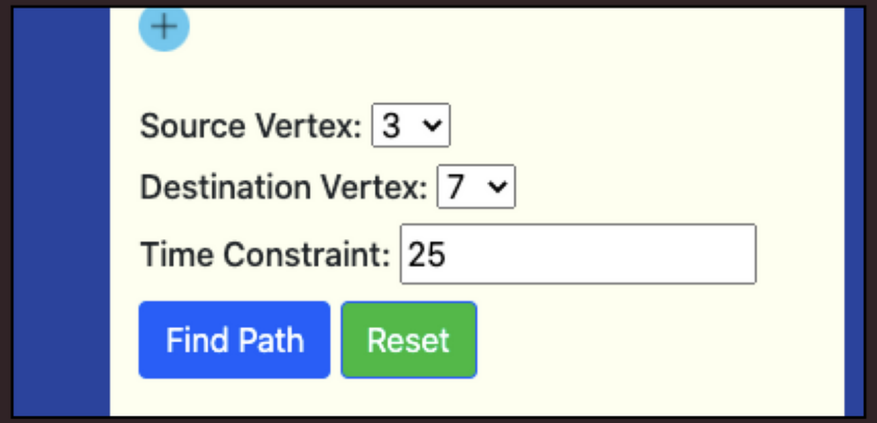
GLOBAL PLANNING ALGO



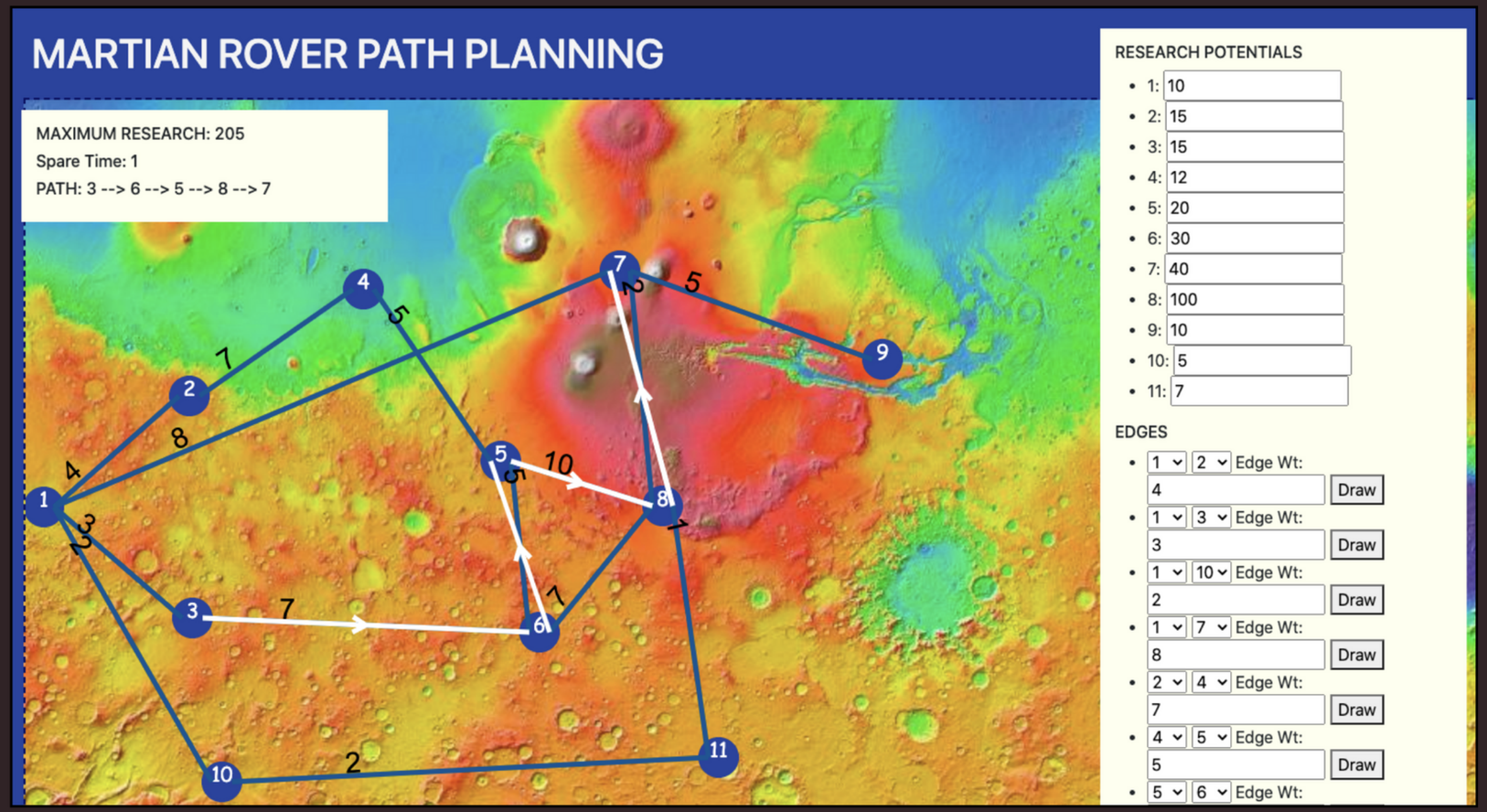
System simulation on our webpage

TEST 1

- INPUT:
  ➢ SRC = 3, DEST = 7,
  TIME CONSTRAINT = 25
- OUTPUT OBTAINED:
  ➢ MAX RESEARCH = 205
  ➢ PATH TO BE FOLLOWED = 3 -> 6 -> 5 -> 8 -> 7
- THE PATH TO BE FOLLOWED HAS BEEN DISPLAYED IN WHITE, AS VISIBLE IN THE IMAGE BELOW.



Input



Output

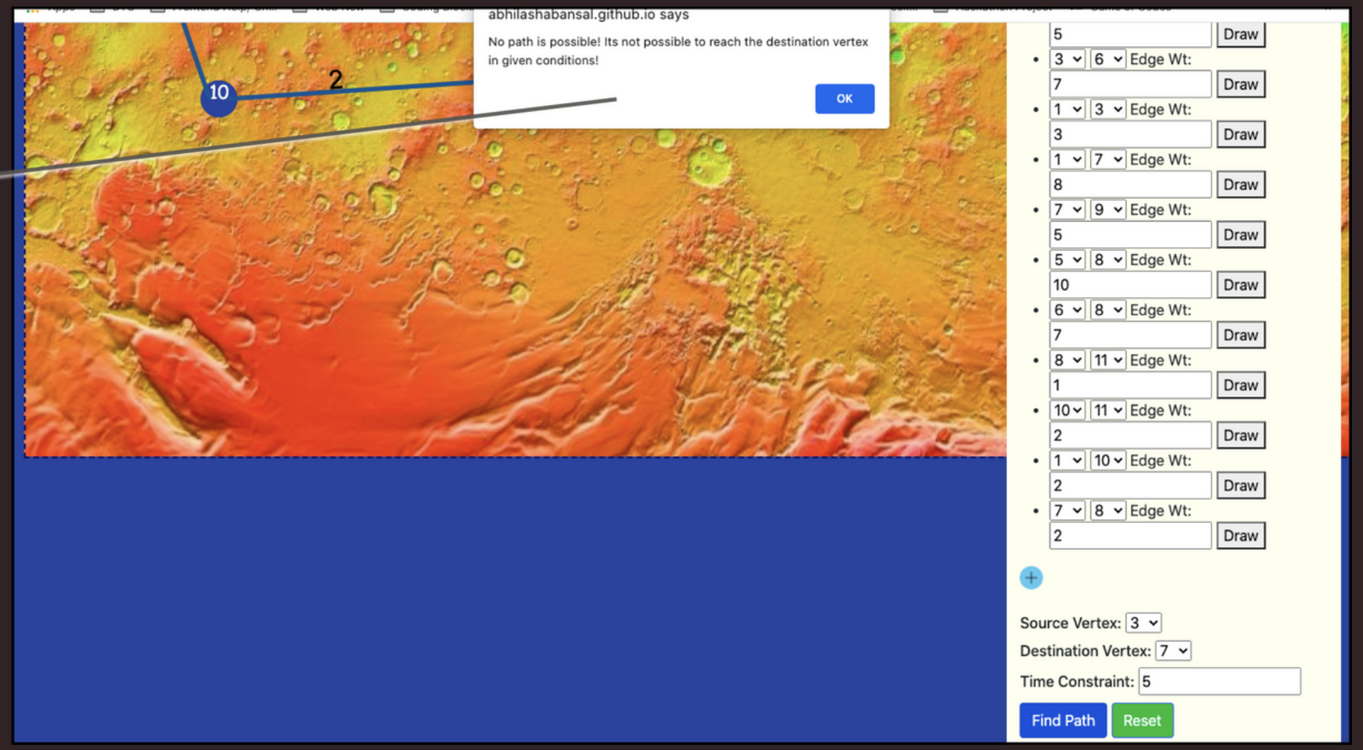
TEST 2

INPUT:
- SRC = 3, DEST = 7, TIME CONSTRAINT = 5

OUTPUT OBTAINED:
- NO PATH POSSIBLE

(THE WEBPAGE GIVES A PROMPT THAT A PATH FROM SOURCE TO DESTINATION VERTEX IS NOT POSSIBLE WITHIN GIVEN CONSTRAINTS.)

Dialogue box telling no path possible.



A STAR ALGORITHM

TEST 1

```
Implementation of A Star Algorithm for NavMars
------------------------------------------------------------------------------
Enter number of rows and columns in grid respectively
9 10
Enter the matrix with 1 for obstacles and 0 for traversable points
0 1 0 0 0 0 1 0 0 0
 0 0 0 1 0 0 0 1 0 0
 0 0 0 1 0 0 1 0 1 0
 1 1 0 1 0 1 1 1 1 0
 0 0 0 1 0 0 0 1 0 1
 0 1 0 0 0 0 1 0 1 1
 0 1 1 1 1 0 1 1 1 0
 0 1 0 0 0 0 1 0 0 0
 0 0 0 1 1 1 0 1 1 0
Enter co-ordinates of source vertex
8 0
Enter co-ordinates of destination vertex
0 0
0 0
------------------------------------------------------------------------------
------------------------------------------------------------------------------
Output Grid
------------------------------------------------------------------------------
Path vertices -> '9'
------------------------------------------------------------------------------
9 1 0 0 0 0 1 0 0 0
9 0 0 1 0 0 0 1 0 0
0 9 0 1 0 0 1 0 1 0
1 1 9 1 0 1 1 1 1 0
0 9 0 1 0 0 0 1 0 1
9 1 0 0 0 0 1 0 1 1
9 1 1 1 1 0 1 1 1 0
9 1 0 0 0 0 1 0 0 0
9 0 0 1 1 1 0 1 1 0
------------------------------------------------------------------------------
------------------------------------------------------------------------------
```

TEST 2

```
Implementation of A Star Algorithm for NavMars
-----------------------------------------------------------------------
Enter number of rows and columns in grid respectively
9 10
Enter the matrix with 1 for obstacles and 0 for traversable points
0 1 0 0 0 0 1 0 0 0
 0 0 0 1 0 0 0 1 0 0
 0 0 0 1 0 0 1 0 1 0
 1 1 0 1 0 1 1 1 1 0
 0 0 0 1 0 0 0 1 0 1
 0 1 0 0 0 0 1 0 1 1
 0 1 1 1 1 0 1 1 1 0
 0 1 0 0 0 0 1 0 0 0
 0 0 0 1 1 1 0 1 1 0
Enter co-ordinates of source vertex
0 0
Enter co-ordinates of destination vertex
8 8
8 8
-----------------------------------------------------------------------
No traversable path from src to destination
-----------------------------------------------------------------------
Program ended with exit code: 0
```

CONCLUSION

The algorithm was simulated and tested on the webpage and the results were found to be accurate.
Hence, this system can be implemented at scale and can be used in martian rovers for a more productive and efficient research system.

The webpage has been deployed on GitHub Pages through the following repository on GitHub:
https://github.com/AbhilashaBansal/ada_project

The source code for all the algorithms have been uploaded at:
https://github.com/its7ARC/martianRoverNavigationSystem