U.T.E.K

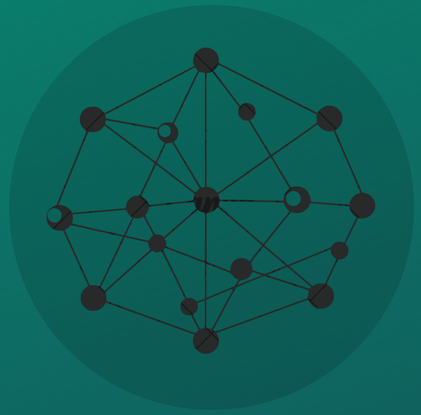
PROGRAMMING COMPETITION TEAM 4

By Krishna, Snehal, Tanay, Parthiv

U.T.E.K

AGENDA FOR TODAY

Problem Statement



Part 1: Graph Construction

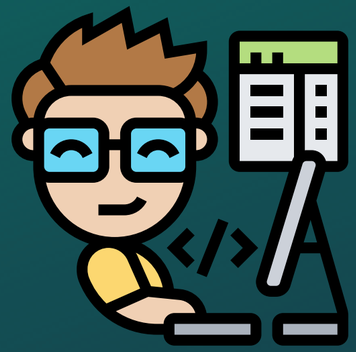
Part 2: Optimal Path

Part 3: Time Constraint

U.T.E.K

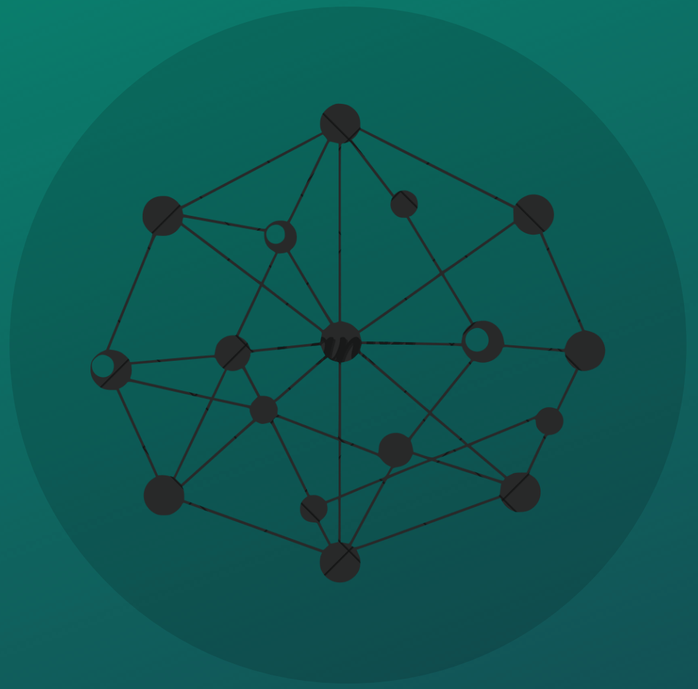
PROBLEM STATEMENT

- Visionary Mayor decides to transform city’s landscape
  - No fossil fuels - Shift to renewable energy sources
  - Utilize clean and efficient energy
  - A greener and sustainable future
- Our Challenge as a team of Programmers:
  - Optimize the city's electricity distribution system
  - Search for an efficient and cost-effective solution



U.T.E.K

PART 1



Graph Construction

U.T.E.K

PART 1: GRAPH CONSTRUCTION

PROBLEM

INPUT

A string that contains pairs of intersections separated by the arrow “->” and comma “,” characters.

TASKS

- Parse through the input string
- Create and output a resulting adjacency matrix/list for the graph

U.T.E.K

PART 1: GRAPH CONSTRUCTION

APPROACH

PARSING THE INPUT

- Remove all the spaces in the input string
- Use comma and arrow operators as delimiter to obtain individual intersection pairs

CREATING ADJACENCY MATRIX

- Create a dictionary of dictionaries of **char : int** as key-value pairs
- E.g - **{‘a’ : {‘b’ : 1, ‘c’ : 0}}** represents that **a->b** exists but **a->c** does NOT

U.T.E.K

PART 1: GRAPH CONSTRUCTION

EXAMPLE

SAMPLE INPUT

a->b, b->c, c->a, a->d

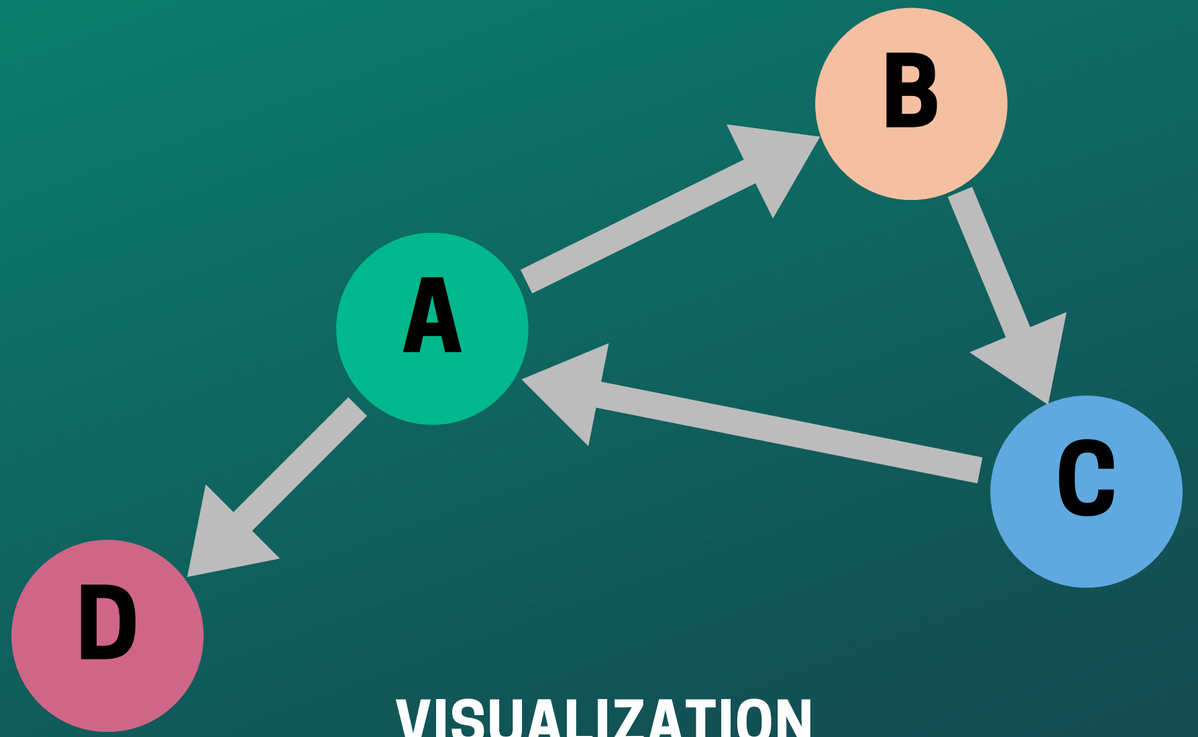
SAMPLE OUTPUT

	a	b	c	d
a	∞	1	∞	1
b	∞	∞	1	∞
c	1	∞	∞	∞
d	∞	∞	∞	∞

U.T.E.K

PART 1: GRAPH CONSTRUCTION

EXAMPLE



VISUALIZATION

U.T.E.K

PART 2

Optimal Path

U.T.E.K

PART 2: OPTIMAL PATH

PROBLEM

INPUT

Starting Intersection, Ending Intersection and a string which denotes pair of intersections along with the cost of fuel (in dollars) required to travel between them

TASKS

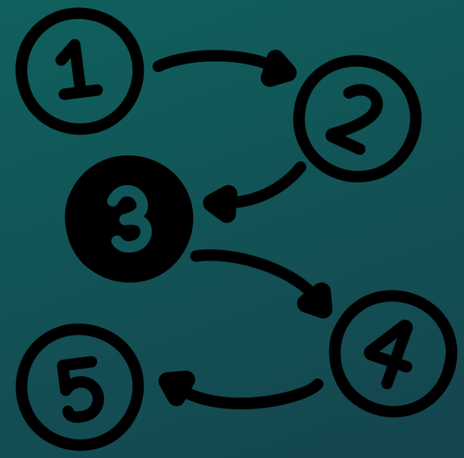
- Parse through the input string to get the cost between intersections
- Find the path from start to end intersection such that every node is visited at least once and it is the most optimal path in terms of fuel cost

U.T.E.K

PART 2: OPTIMAL PATH

APPROACH

- The solution we developed involves using iterative backtracking.
- We remove the first and last node from the graph and iteratively check every possible permutation of the remaining nodes.
- Then, we find the cost of traversing these paths and select the one with the lowest cost.



U.T.E.K

PART 2: OPTIMAL PATH

EXAMPLE

SAMPLE INPUT

a->b ($4), b->c ($5), c->d ($3), d->b ($4), a->c ($4), d->a ($1)

Given that the starting intersection is ‘a’, ending intersection is ‘b’.

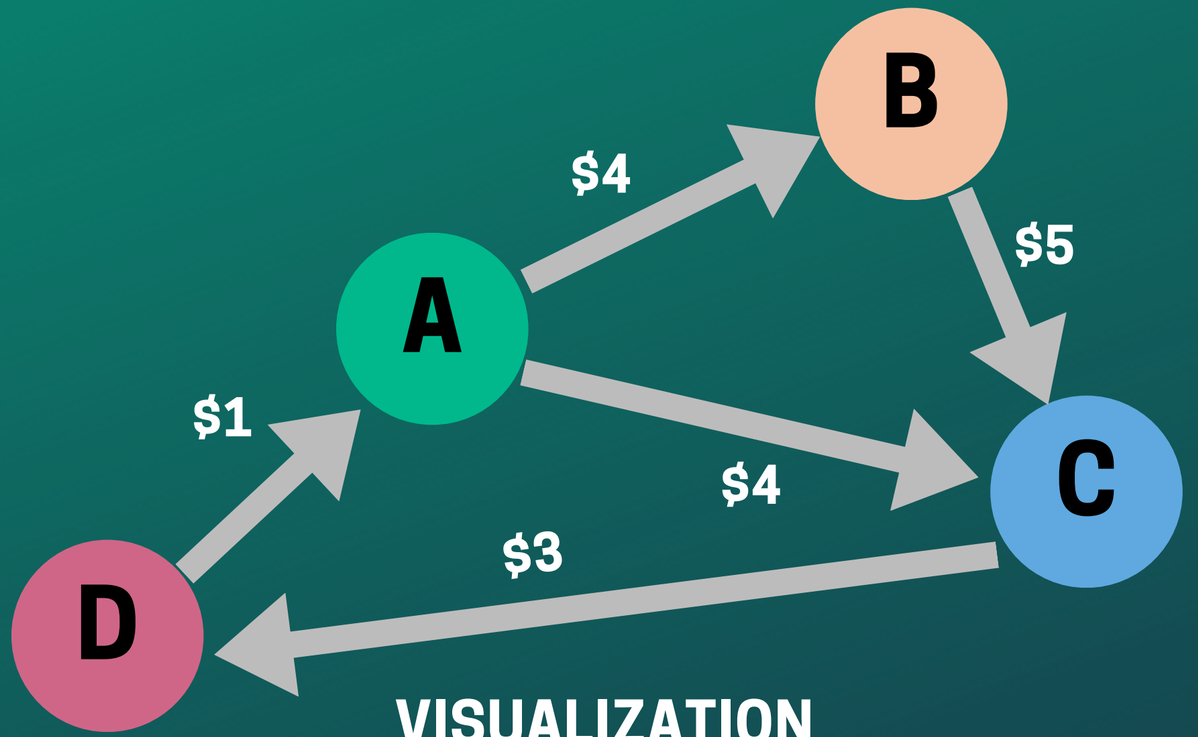
SAMPLE OUTPUT

- **Best path:** a->c, c->d, d->b
- **Cost:** $11

U.T.E.K

PART 2: OPTIMAL PATH

EXAMPLE



VISUALIZATION

U.T.E.K

PART 3

Time Constraint

U.T.E.K

PART 3: TIME CONSTRAINT

PROBLEM

INPUT

Starting Intersection, Ending Intersection, Maximum Time Limit and a string which denotes pair of intersections along with the cooldown between intersections and the cost of fuel (in dollars) required to travel between them

TASKS

- Parse through the input string to get the fuel cost and cooldown time between intersections
- Find a path from start to end intersection such that every intersection is visited at least once and the cost of fuel used is minimum. Also, the sum of all the cooldowns in the path must be less than or equal to the Max Time Limit.

U.T.E.K

PART 3: TIME CONSTRAINT

APPROACH

- Parse the string into two tables (using nested dictionaries):
  - Adjacency matrix (tracking cost from going one node to another)
  - Cooldown matrix (tracking cooldown period from going to one node from another)
- Implement iterative backtracking to find a solution that meets the aforementioned criteria (sum of the cooldowns lower than the max time limit, and minimizing total fuel cost)

U.T.E.K

PART 3: TIME CONSTRAINT

EXAMPLE

SAMPLE INPUT

a->b ($4, 1 min cooldown) , b->c ($5, 1 min cooldown), c->d ($3, 3 min cooldown) , b->d ($7, 3 min cooldown), a->c ($4, 3 min cooldown), d->a ($1, 4 min cooldown)

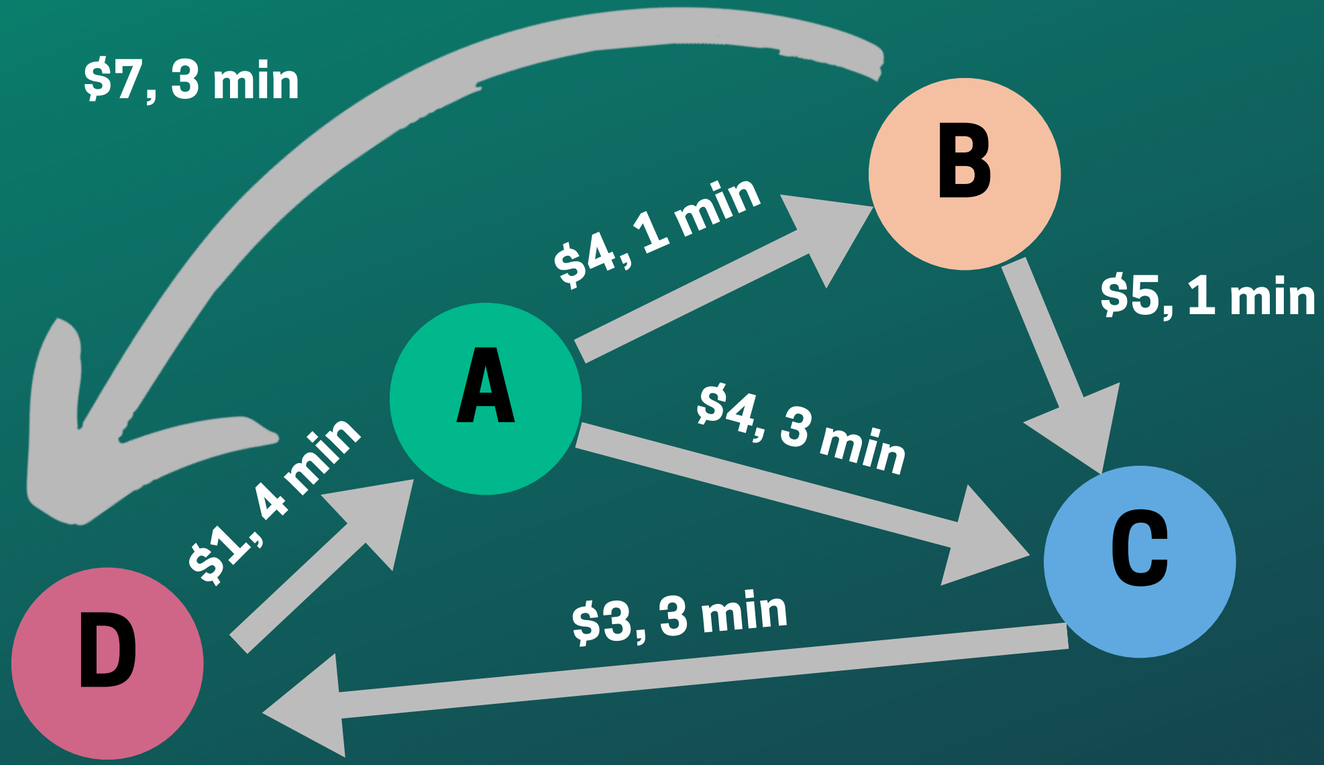
Given that the starting intersection is ‘a’, ending intersection is ‘b’ and the max cooldown period is 5 minutes

SAMPLE OUTPUT

- **Best path:** a->b, b->c, c->d
- **Cost:** $12
- **Time:** 5 min

U.T.E.K

PART 3: TIME CONSTRAINT



U.T.E.K

CONCLUSION

Teamwork and Planning:

- Understanding the problem collectively
- Strategizing and work distribution among us

Learnings and exploring the solution space:

- Dynamic Programming
- Dijkstra and A* Algorithm
- NP-Hard Problem

U.T.E.K

THANK YOU

FROM PROGRAMMERS