CS6023 GPU Course Project

Submitted by: **N K Sagar Reddy**
, P Bharat Simha Reddy

Roll no: **CS18B029, CS18B051**

Department of Computer science and Engineering

May 14, 2021

1 Idea

- We would like to find the shortest path between any 2 given nodes in a dynamically changing grid in an efficient manner.
- The idea is to parallelize the algorithm used(Dynamic A*) and also change the computed data instead of recomputing it from the start everytime the grid changes.
- This algorithm is mainly useful in the gaming industry where the environment changes continuously and when we need to find the optimal path to the destination.

2 Project info

- We have parallelized sequential Dynamic A* pathfinding algorithm in CUDA.
- Key Concepts used: Implementation of Priority Queue using linked list on a GPU , Dynamic Parallelism .
- We have an input grid,where -1s represent blockages , other non negative values represent the cost added to the path which passes through that particular location.
- We also dynamically add edges/values to this grid (as given in the input file) in between 2 shortest path computations.
- There are 2 types of queries given in the input file : Compute(id = 7) and Add(id = 3).
- For each Compute Query ,we find the shortest path from the already given source and destination at that point of time and print it.
- For each Add query,we add one or more edges to the given grid with their values/costs as mentioned in the input file.

3 Input Format

- The first line contains n and m(representing the number of rows and columns in the grid)
- The next n lines contain m integers each,which represent the Grid.
- Below the grid, the next line contains 4 integers: srcx,srcy,destx,desty , which represent the locations of the source and destination in the given grid.
- The next line contains Q which is the number of queries. Q queries follow.
- **Query Type - 1:** The query format is ”7”. we have to compute and print the shortest path upon scanning this query.
- **Query Type - 2:** The query format is ”3 e”,e lines follow.(e edges to be added)
- Each of the edge to be added is of the ”x y val” format,which says to update the grid[x][y] to the value ”val”.

4 Challenges Faced

- We implemented A* first,then we implemented Dynamic A* without recomputing some key data and just making required modifications after each edge addition. So we had to write a function/kernel for each edge addition,which takes care of certain implementation aspects such as setting the parent of the given location and updating the grid after the change has been made.
- To increase the efficiency, we have used the concept of Dynamic parallelisation,which ensures kernel within kernel calls. This helped us to maintain a priority queue inside the GPU,
  we had to implement the priority queue using linked list based functions which were accessible to the kernel(device functions) as the kernel cant call global functions.