# Project 2: Prim's algorithm implementation using indexed priority queue

## CS 5V81.001: Implementation of data structures and algorithms

Sreesha Nagaraj (sxn146630)

## Prim's Algorithm:

- Input: A non-empty connected weighted graph with vertices V and edges E (the weights can be negative).
- Initialize: $V_{new}$ = {x}, where x is an arbitrary node (starting point) from V, $E_{new}$ = {}
- Repeat until $V_{new}$ = V:
  - Choose an edge {u, v} with minimal weight such that u is in $V_{new}$ and v is not (if there are multiple edges with the same weight, any of them may be picked)
  - Add v to $V_{new}$, and {u, v} to $E_{new}$
- Output: $V_{new}$ and $E_{new}$ describe a minimal spanning tree

## Implementation

- Indexed Priority queue represents binary Min-heap structure. Every vertex in graph is an element in a priority queue.
- Index value gives the access to the element in the queue in O (1) time.
- Main functionality includes decrease key, percolate up, percolate down, DeleteMin for priority queue
- Decrease key will be called in prim's algorithm when node key of the vertex is reduced (In this implementation key is weight that is associated with vertex).
- Decrease key restores heap order property after the priority of x has decreased by calling percolate up function.
- DeleteMin function removes the 1st node of the heap (minimum weight node) and then call Percolate down because Heap order may be violated.

## Input

- http://www.utdallas.edu/~rbk/teach/now/projects/proj2/data-prim.zip
- http://www.utdallas.edu/~rbk/teach/now/projects/proj2/bigdata-prim.zip

## Output

| File | Vertices count | Edges count | Weight of MST | Running Time (To calculate MST) |
|---|---|---|---|---|
| Prim1.txt | 50 | 140 | 84950 | 2 ms |
| Prim2.txt | 100 | 284 | 110419 | 2ms |
| Prim3.txt | 200 | 580 | 153534 | 6ms |
| Prim-ck.txt | 100000 | 299971 | 3384476 | 162ms |