**Comparing the performance of Prim1 and Prim2**

**Prim1**- **priority queue of edges; using Java's priority queues**

Here on running the big-input.txt File on prim1 it took around 32-35 seconds whereas **Prim2**->**priority queue of vertices, using indexed heaps** took just 26-28 seconds.

This is because when E is very large it’s better to use indexed-Heaps for priority queue as we will have many nodes in the queue. It will take time to find min/remove min in 0(n)/0 (n) times whereas heap only takes 0(1)/log (n) time.

But if the E is not large pretty small, we will have very few nodes in the queue and thus to find the min, and remove it from the array will not need a lot of operation in this case. In this using a heap will not be necessary and it will be slower than array due to operations needed to build the Heap.

**Time in Msec**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | Dijkstra's shortest Path | Prim-using PQ of Vertex-Indexed heaps | Prim-using PQ of edges | Output of prim | Output of Dijkstra's shortest Path |
| G4-big.txt | 636 | 600 | 4392 | 10000 | 4 |
| G1.txt | 3 | 4 | 3 | 84950 | 12020 |
| G2.txt | 4 | 5 | 5 | 110419 | 9106 |
| G3.txt | 5 | 5 | 7 | 153534 | 10672 |

**Therefore By comparing the above values on Large Input Prim2 – of vertex-Indexed heaps runs faster On Dense Graphs than Prim1-pq of Edges.**

**Classes in Project**

**Timer.java**-> This contains as the name suggests how much time the program took to execute.

**Driver.java** -> This is the main program that is used to run Prim2 (priority queue of vertices, using indexed heaps), Dijkstra's algorithm for shortest paths using indexed heaps.

**BinaryHeap.java** -> This class contains basic implementations of Java’s Priority queue using Binary Heap DS.

**MST.Java** -> This class contains two flavors of MST one using java’s Priority queue and other using Indexed Heaps.

**ShortestPath.Java** -> This class contains Implementations of Dijkstra's algorithm for shortest paths using indexed heaps.

All other classes are self-Explanatory.