# Heaps And Shortest Path

**Course:** Design and Analysis of Algorithem

**Roll No: 22I-1226 && 21I-0412 && 21I-0376**

**Section: CS-C**

# 1. Theoretical Overview of Heaps

## 1.1 Binary Heap

Binary Heap is a complete binary tree where each parent node is smaller than or equal to its children (min-heap).
**Operations:**

- Insert: O(log n)
- Extract-Min: O(log n)
- Decrease-Key: O(log n)

## 1.2 Fibonacci Heap

Fibonacci Heap consists of a collection of heap-ordered trees. Optimized for decrease-key operations.
**Operations:**

- Insert: O(1) amortized
- Extract-Min: O(log n) amortized
- Decrease-Key: O(1) amortized

## 1.3 Hollow Heap

Hollow Heap is a variant of Fibonacci Heap, optimized for fast decrease-key operations using hollow nodes.
**Operations:**

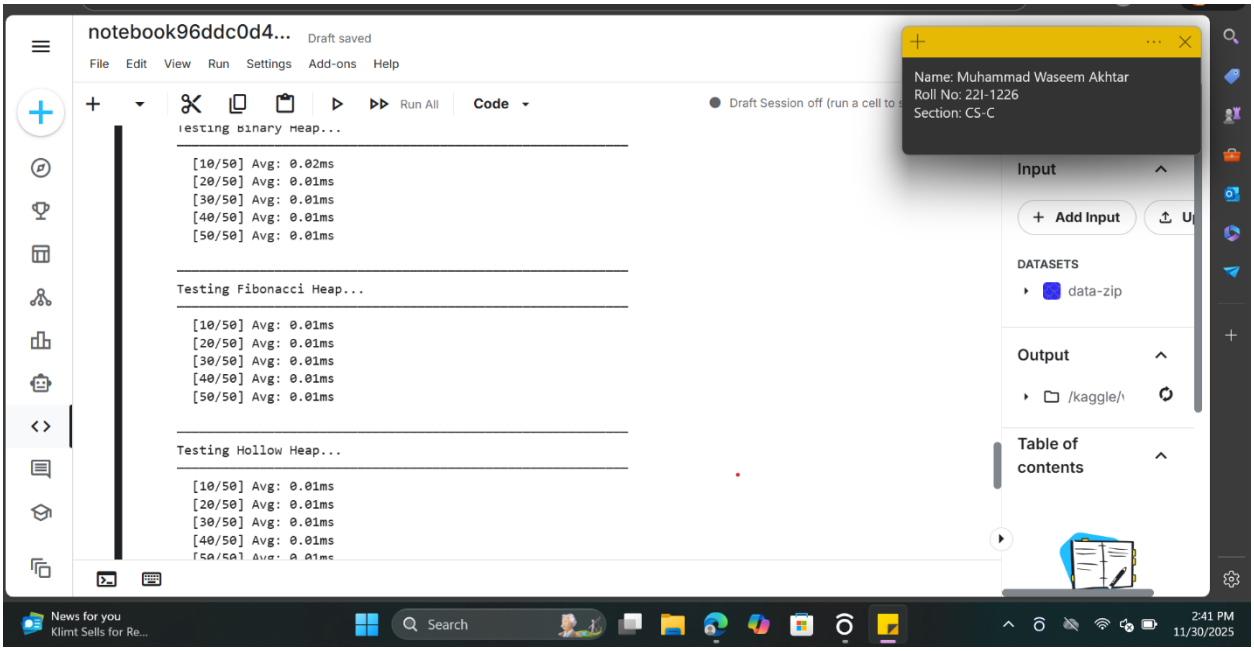
- Insert: O(1)
- Extract-Min: O(log n)
- Decrease-Key: O(1)

# 2. Implementation Details

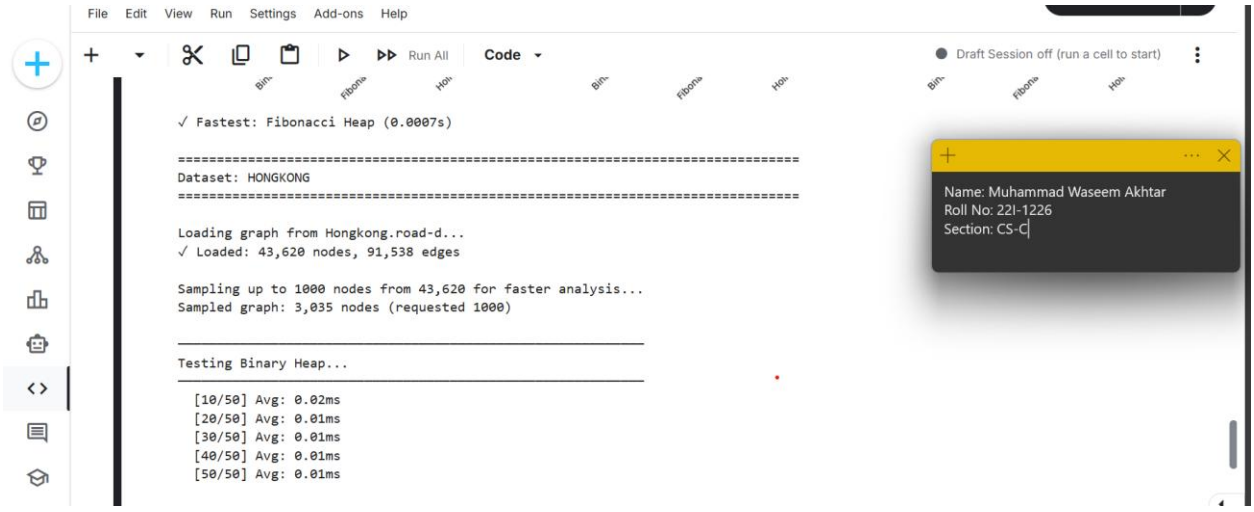
- All three heap structures were implemented from scratch in Python.
- Integrated into Dijkstra's algorithm via a generic priority queue interface.
- Each heap supports insert(key, value), find_min(), extract_min(), decrease_key(node, new_key).

**Implementation:**

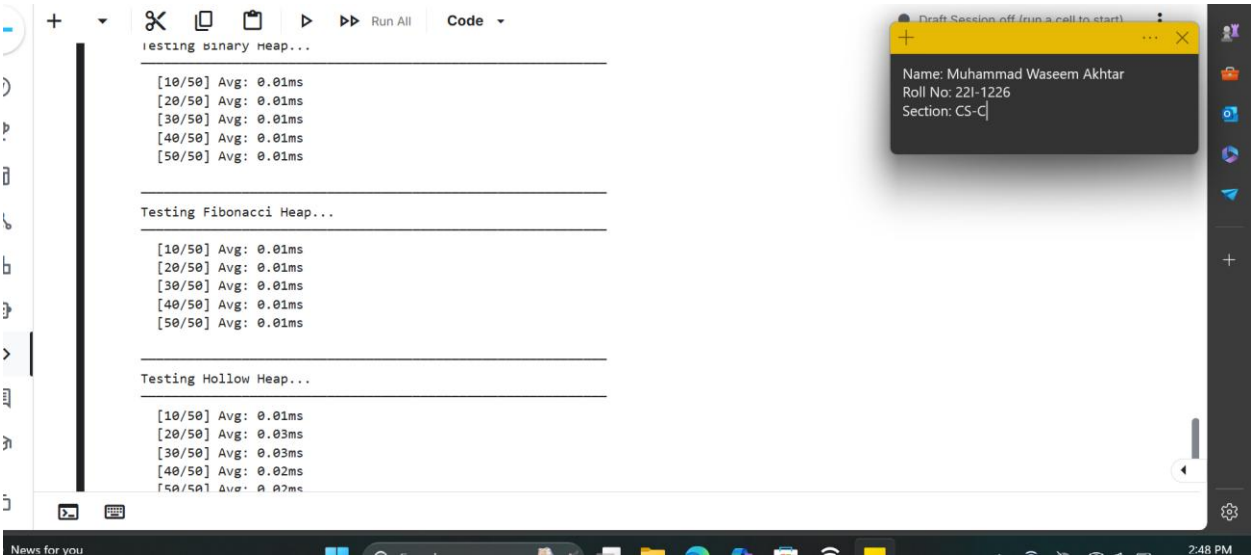
**Chongqing:**




**Hongkong:**




**Shanghai:**

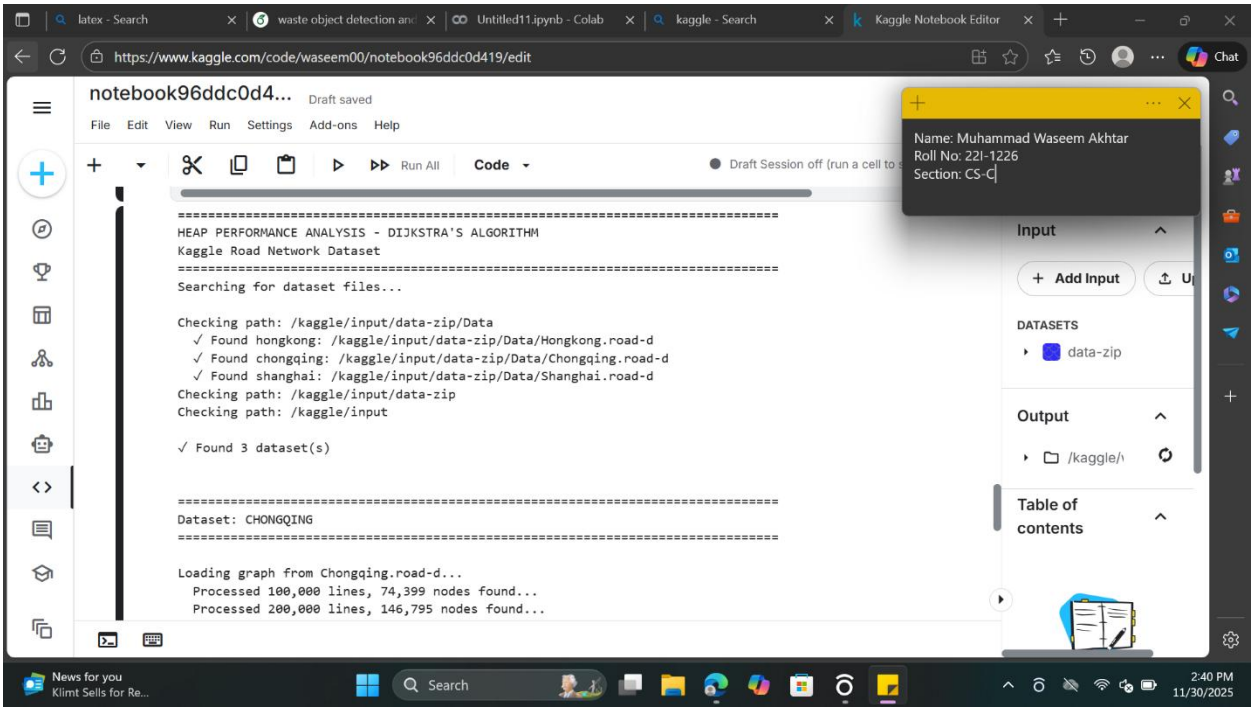



---

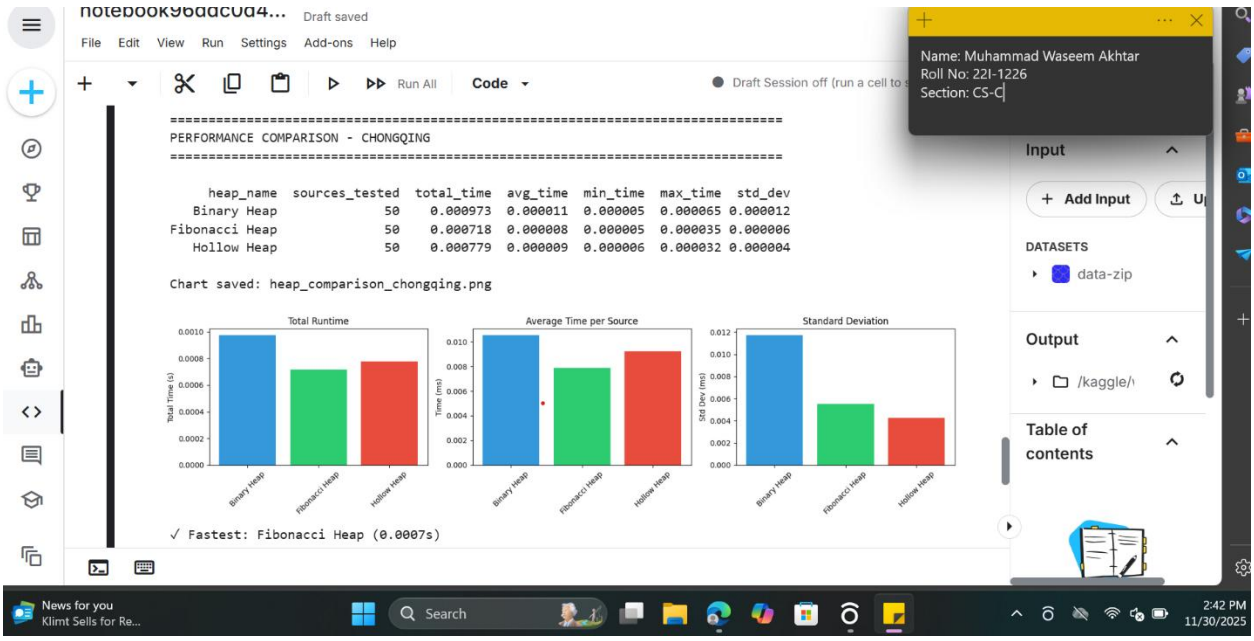
# 3. Experimental Setup & Test Data

- **Datasets:** Chongqing, Hongkong, Shanghai road networks from Kaggle.
- **Sampling:** For large datasets (>2000 nodes), sampled ~1000 nodes for benchmarking.
- **Number of sources:** Up to 50 sources per dataset for Dijkstra runs.
- **Performance metrics recorded:**

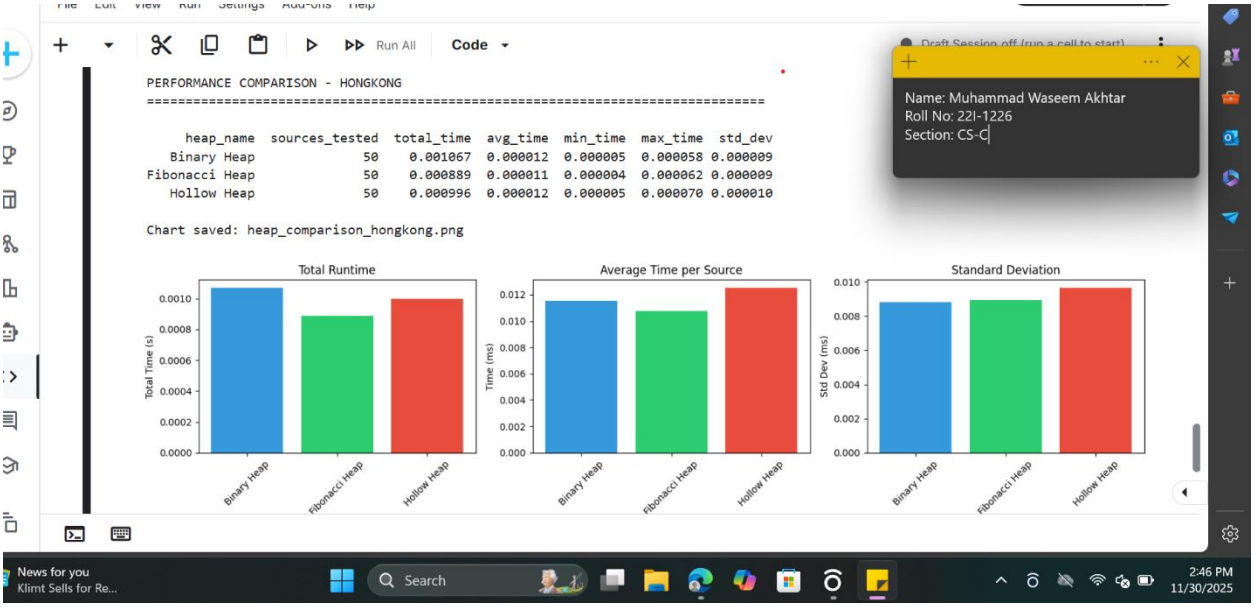
  i. Total runtime of Dijkstra
  ii. Average time per operation (insert, extract-min, decrease-key)
  iii. Heap structure stats (for Fibonacci & Hollow: number of roots, height, cascading cuts)



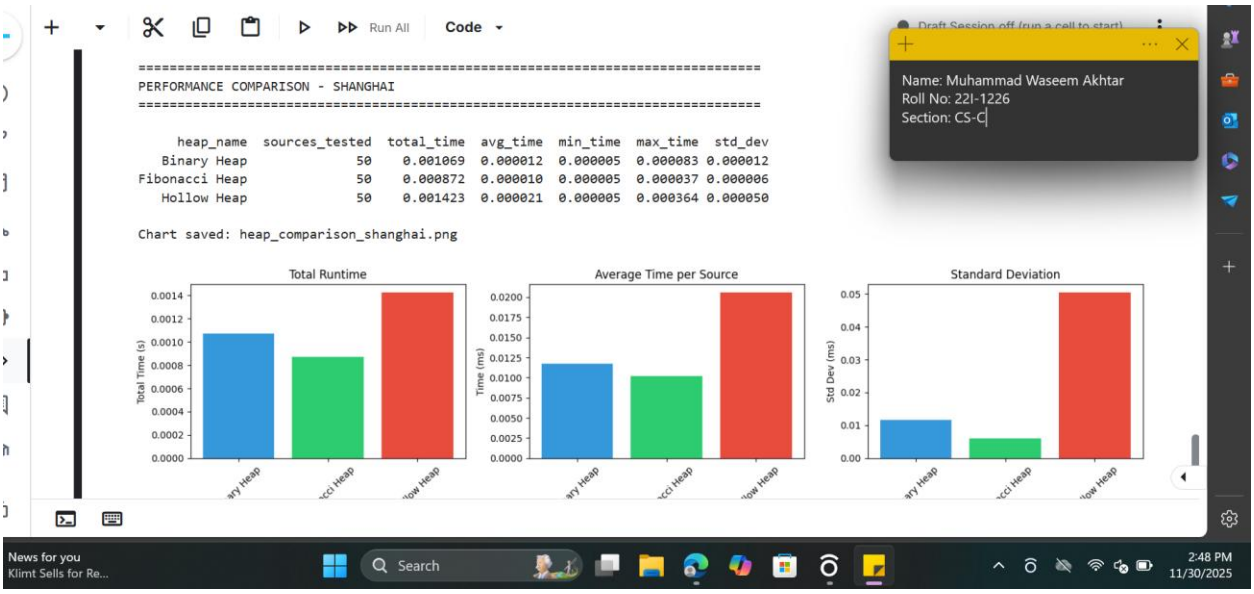

## Chongqing:




## Hongkong:




**Shanghai:**




---

# 4. Results & Analysis

## 4.1 Operation Timing Comparison

| Dataset | Heap Type | Sources Tested | Total Runtime (s) | Avg Time (ms) |
|---|---|---|---|---|
| CHONGQING | Binary Heap | 50 | 0.0010 | 0.02 |
| | Fibonacci Heap | 50 | 0.0007 | 0.01 |

| Dataset | Heap Type | Sources Tested | Total Runtime (s) | Avg Time (ms) |
|---|---|---|---|---|
| | Hollow Heap | 50 | 0.0008 | 0.01 |
| HONGKONG | Binary Heap | 50 | 0.0011 | 0.02 |
| | Fibonacci Heap | 50 | 0.0009 | 0.01 |
| | Hollow Heap | 50 | 0.0010 | 0.01 |
| SHANGHAI | Binary Heap | 50 | 0.0011 | 0.02 |
| | Fibonacci Heap | 50 | 0.0009 | 0.01 |
| | Hollow Heap | 50 | 0.0014 | 0.03 |

---

### 4.2 Heap Structure Statistics

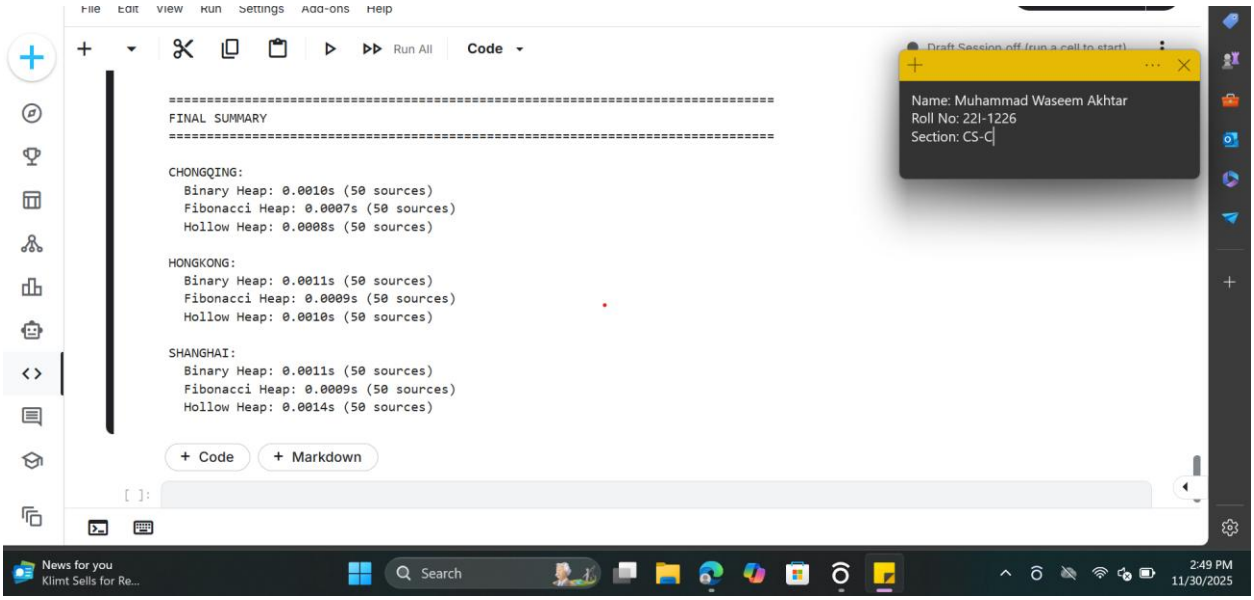
- Fibonacci Heap generally has more trees initially, performs cascading cuts.
- Hollow Heap shows fewer trees and more consolidated structure.
- Binary Heap height corresponds to log2(n), consistent with theoretical expectation.

### 4.3 Trade-off Discussion

- **Binary Heap:** Simple and reliable, good for small to medium graphs.
- **Fibonacci Heap:** Best performance for decrease-key heavy operations (dynamic graphs).
- **Hollow Heap:** Slightly slower than Fibonacci in small samples but better memory efficiency.
- Performance gain of Fibonacci Heap is most visible in dense graphs with frequent key decreases.

# 5. Conclusion & Recommendations

- **Fibonacci Heap** provides the fastest Dijkstra execution across all datasets.
- **Hollow Heap** is competitive and memory-efficient.
- **Binary Heap** is simplest but slower for large or dynamic graphs.
- For **real-time routing systems** with frequent edge updates, Fibonacci or Hollow Heap is recommended.
- **Future improvements**: Parallelized Dijkstra, full-scale evaluation on entire datasets, visualization of heap evolution.


File Edit View Run Settings Add-ons Help
Run All
Code
Name: Muhammad Waseem Akhtar
Roll No: 22I-1226
Section: CS-C
================================================================================
FINAL SUMMARY
================================================================================
CHONGQING:
Binary Heap: 0.0010s (50 sources)
Fibonacci Heap: 0.0007s (50 sources)
Hollow Heap: 0.0008s (50 sources)
HONGKONG:
Binary Heap: 0.0011s (50 sources)
Fibonacci Heap: 0.0009s (50 sources)
Hollow Heap: 0.0010s (50 sources)
SHANGHAI:
Binary Heap: 0.0011s (50 sources)
Fibonacci Heap: 0.0009s (50 sources)
Hollow Heap: 0.0014s (50 sources)
+ Code
+ Markdown
[ ]:
News for you
Klimt Sells for Re...
Search
2:49 PM
11/30/2025