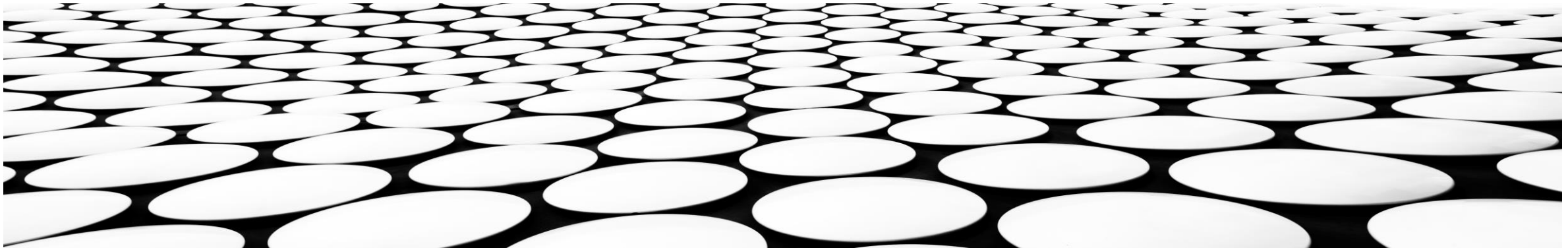
Materi Tambahan Graph (kompleksitas Algoritma)

Suryana Setiawan, PhD



Traversal: Algoritma BFS dan DFS

- Graph memiliki n vertex dan m edges (average degree D = m/n):
  - Untuk BDS: n iterasi yang di dalamnya terjadi operasi dequeue dan iterasi adjacency sebanyak D vertex berikutnya:
    - O(n(1+D)) = O(n + m)
  - Untuk DFS: n iterasi yang di dalamnya secara implisit terjadi 1 pop dan iterasi adjacency sebanyak D vertex berikutnya:
    - O(n(1+D)) = O(n + m)

- Jika m merupakan fungsi dari n misalnya m = α n^2 maka dapat menjadi O(n^2)

Shortest Path: Algoritma Dljkstra

- Graph memiliki n vertex dan m edges (D = m/n):
  - Table Dist **array biasa (sikuensial)**:
    - $O(n (n + D)) = O(n^2 + m) = O(n^2)$ karena $n \sim m << n^2$
  - Table Dist **Binary MinHeap priority queue**:
    - $O(n (\log n + D \log n)) = O(n \log n + m \log n) = O((n + m) \log n)$
  - Table Dist **Fibonacci Heap priority queue**: $O(m + n \log n)$
    - Memungkinkan update heap terjadi O(1) (dari adjacent vertices) [*Fredman, Michael Lawrence; Tarjan, Robert E. (1984). Fibonacci heaps and their uses in improved network optimization algorithms. 25th nAnnual Symposium on Foundations of Computer Science. IEEE. pp. 338–346. doi:10.1109/SFCS.1984.715934.*]
- Jika m merupakan fungsi dari n misalnya $m = \alpha n^2$ maka versi Binary Minheap menjadi $O(n^2 \log n)$ dan Fibonacci Heap menjadi $O(n^2)$

Topological Sort

- Graph memiliki n vertex dan m edges (D = m/n):
  - In-degree computation:
    - Adjacency list: O(m).
    - Adjacency matrix: $O(n^2)$
  - Dalam n kali iterasi, masing-masing:
    - Mencari in-degree yang 0: O(n)
    - Update in-degree dari adjacent
      - Adjacency list: O(D) = O(m/n)
      - Adjacency mtrix: O(D) = O(n)
- Kompleksitas total:
  - Adjacency list (m upperbounded oleh n^2):
    - $O(m) + O(n\ (n + m/n)) = O(m + n^2) = O(n^2)$
  - Adjacency matrix:
    - $O(n^2)$

MST: Algoritma Prim

- Analisis mirip Dijkstra, dalam setiap dari n iterasi:
  - Mendapatkan table sisi minimum
  - Mengupdate table sisi sebanyak D
- Dengan **table array biasa (sikuensial)**:
  - $O(n\ (n + D)) = O(n^2 + nD) = O(n^2+m) = O(n^2)$
- Dengan **Binary Heap**:
  - $O((n+m) \log n)$
- Dengan **Fibonacci Heap**:
  - $O(m + n \log n)$

MST: Algoritma Kruskal

- Sorting edgelist (jika belum sorted):
  - $O(m \log n^2) = O(m \log n)$
- Inisialisasi MST = sebagai kumpulan disjoint sets yang masing-masing hanya berisi 1 vertex.
- Iterasi dari m sisi, setiap sisi:
  - Periksa edge terpendek (x, y) membentuk siklik
    - Dengan memeriksa apakah x dan y dari dua disjoint set berbeda?
      - Jika ya, merge kedua disjoint → operasi merge O(log n)
    - Else (sudah dalam disjoint set yang sama): do nothing
- Kompleksitas total (sorting dan merging):
  - $O(m \log n + m \log n) = O (m \log n)$