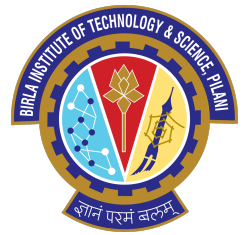
CS F364: Design & Analysis of Algorithms

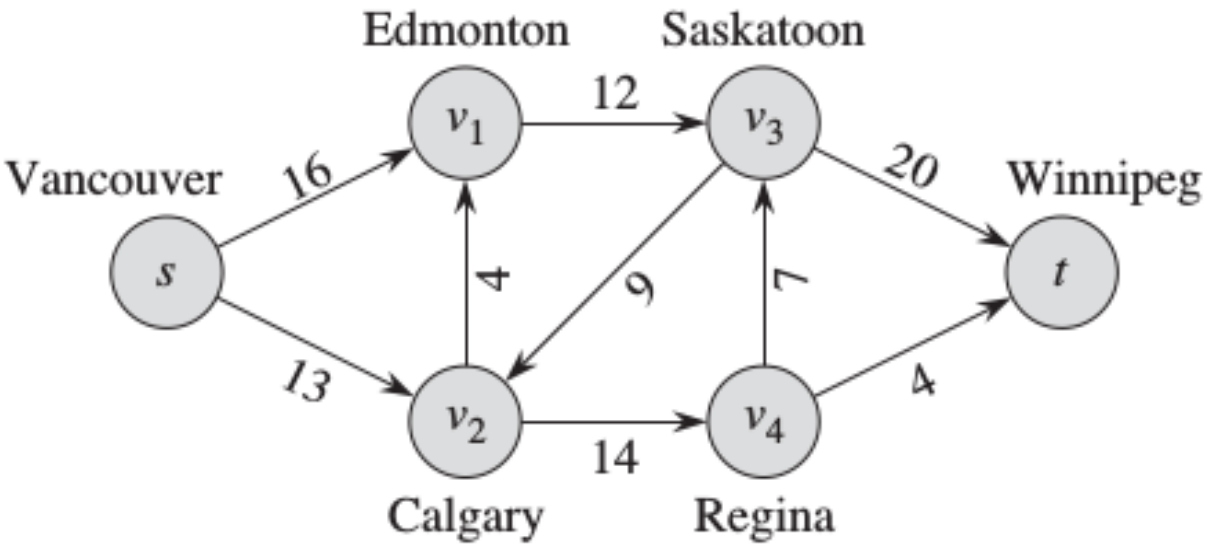
Lecture-kt18: Maximum Flow



Dr. Kamlesh Tiwari,
Assistant Professor,
Department of Computer Science and Information Systems,
BITS Pilani, Rajasthan-333031 INDIA

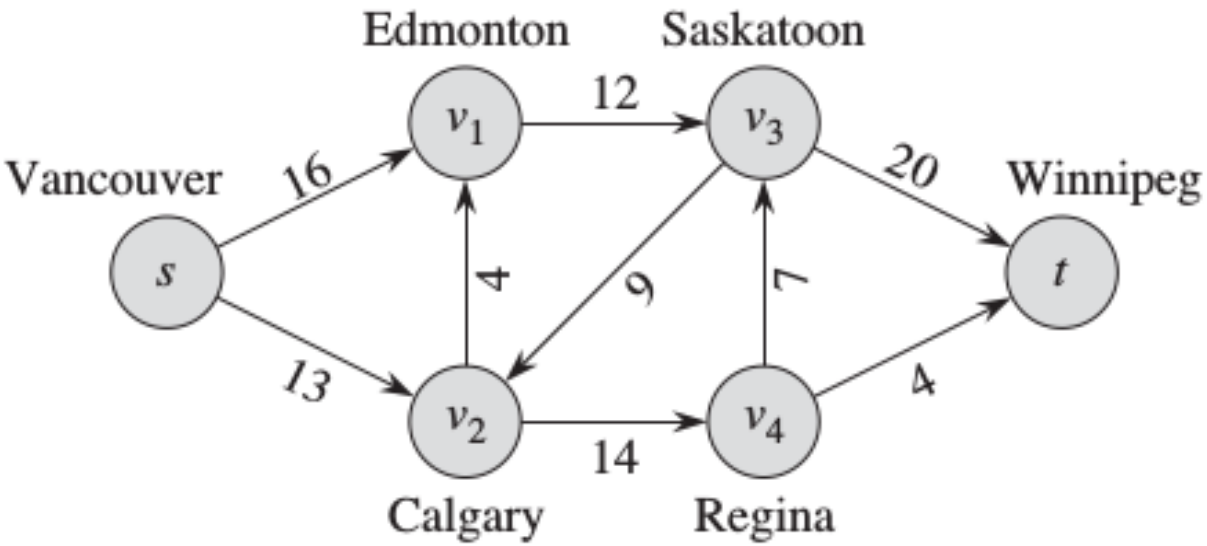
Mar 14, 2017 (Campus @ BITS-Pilani Jan-May 2017)

Maximum Flow



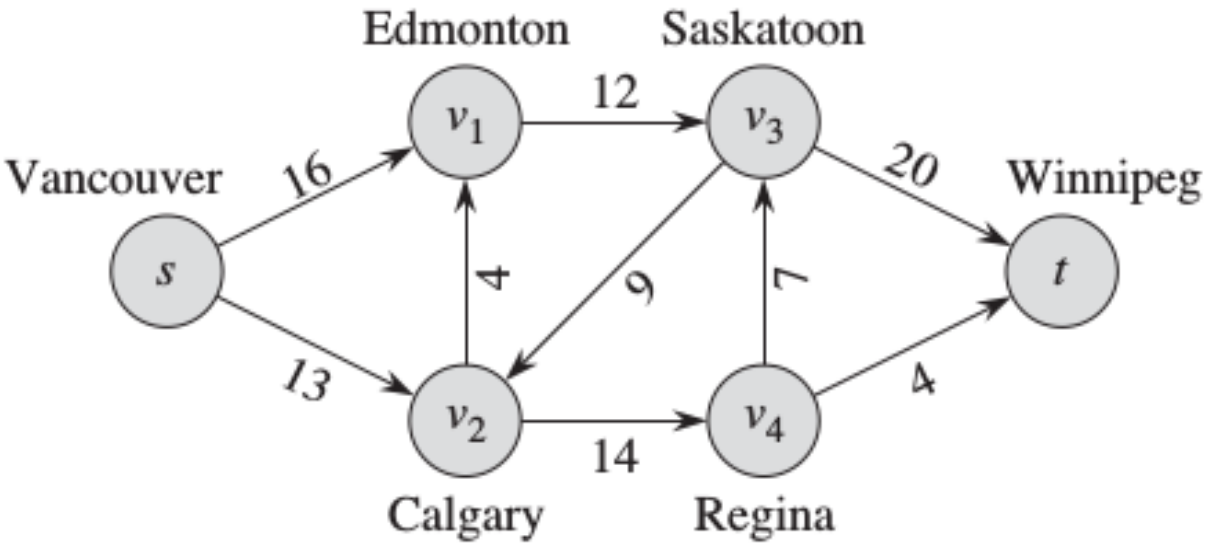
- Let directed edge be a conduit and weight represents its capacity in terms of material that can flow through it

Maximum Flow



- Let directed edge be a conduit and weight represents its capacity in terms of material that can flow through it
- If $(u, v) \in E$ then $(v, u) \notin E$

Maximum Flow



- Let directed edge be a conduit and weight represents its capacity in terms of material that can flow through it
- If $(u, v) \in E$ then $(v, u) \notin E$
- Find the Maximum Flow in the network from *Vancouver* to *Winnipeg*

Maximum Flow

- Let $c(u, v)$ be the capacity of flow that is **nonnegative.** It is zero if no edge between node u and v

Maximum Flow

- Let $c(u, v)$ be the capacity of flow that is **nonnegative**. It is zero if no edge between node u and v
- Let $G = (V, E)$ be a flow network with a capacity function c. Let s be the **source** of the network, and let t be the **sink**. A flow in G is a real-valued function $f : V \times V \rightarrow \mathbb{R}$

Maximum Flow

- Let $c(u, v)$ be the capacity of flow that is **nonnegative**. It is zero if no edge between node u and v
- Let $G = (V, E)$ be a flow network with a capacity function c. Let s be the **source** of the network, and let t be the **sink**. A flow in G is a real-valued function $f : V \times V \rightarrow \mathbb{R}$ that satisfies the following two properties:

Maximum Flow

- Let $c(u, v)$ be the capacity of flow that is **nonnegative.** It is zero if no edge between node u and v
- Let $G = (V, E)$ be a flow network with a capacity function c. Let s be the **source** of the network, and let t be the **sink.** A flow in G is a real-valued function $f : V \times V \rightarrow \mathbb{R}$ that satisfies the following two properties:
  - **Capacity constraint:** For all $u, v \in V$, we require

$$0 \leq f(u, v) \leq c(u, v)$$

Maximum Flow

- Let $c(u, v)$ be the capacity of flow that is **nonnegative.** It is zero if no edge between node u and v
- Let $G = (V, E)$ be a flow network with a capacity function c. Let s be the **source** of the network, and let t be the **sink.** A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:
  - **Capacity constraint:** For all $u, v \in V$, we require

$$0 \leq f(u, v) \leq c(u, v)$$

  - **Flow conservation:** For all $u \in V - \{s, t\}$, we require

$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$$

where $f(u, v)$ is flow from vertex u to vertex v.

Maximum Flow

- The value $|f|$ of **flow** is defines as

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$$

Maximum Flow

- The value $|f|$ of **flow** is defines as

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$$

- In the maximum-flow problem, we are given a flow network G with source s and sink t, and we wish to find a flow of maximum value.

Maximum Flow

- The value $|f|$ of **flow** is defines as

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$$

- In the maximum-flow problem, we are given a flow network G with source s and sink t, and we wish to find a flow of maximum value.
- **Antiparallel edges:** are edges (u, v) and (v, u)

Maximum Flow

- The value $|f|$ of **flow** is defines as

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$$

- In the maximum-flow problem, we are given a flow network G with source s and sink t, and we wish to find a flow of maximum value.
- **Antiparallel edges:** are edges (u, v) and (v, u)
- Multiple source and sink can be handled by adding supersource and supersink

Ford-Fulkerson method

- Depends on three important ideas: 1) residual networks, 2) augmenting paths, and 3) cuts.

Ford-Fulkerson method

- Depends on three important ideas: 1) residual networks, 2) augmenting paths, and 3) cuts.

Algorithm 2: Ford-Fulkerson-Method(G,s,t)

1 Initialize f to 0

Ford-Fulkerson method

- Depends on three important ideas: 1) residual networks, 2) augmenting paths, and 3) cuts.

Algorithm 3: Ford-Fulkerson-Method(G,s,t)

---

1 Initialize f to 0
2 **while** *there exists an augmenting path p in the residual network* G_f **do**

Ford-Fulkerson method

- Depends on three important ideas: 1) residual networks, 2) augmenting paths, and 3) cuts.

Algorithm 4: Ford-Fulkerson-Method(G,s,t)

```
1 Initialize f to 0
2 while there exists an augmenting path p in the residual network G_f
  do
3 |   augment flow f along p
```

Ford-Fulkerson method

- Depends on three important ideas: 1) residual networks, 2) augmenting paths, and 3) cuts.

Algorithm 5: Ford-Fulkerson-Method(G,s,t)

```
1 Initialize f to 0
2 while there exists an augmenting path p in the residual network G_f
  do
3     augment flow f along p
4 return f
```

Ford-Fulkerson method

- Depends on three important ideas: 1) residual networks, 2) augmenting paths, and 3) cuts.

Algorithm 6: Ford-Fulkerson-Method(G,s,t)

```
1 Initialize f to 0
2 while there exists an augmenting path p in the residual network G_f
  do
3     augment flow f along p
4 return f
```

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{otherwise} \end{cases}$$

Ford-Fulkerson method

- Depends on three important ideas: 1) residual networks, 2) augmenting paths, and 3) cuts.

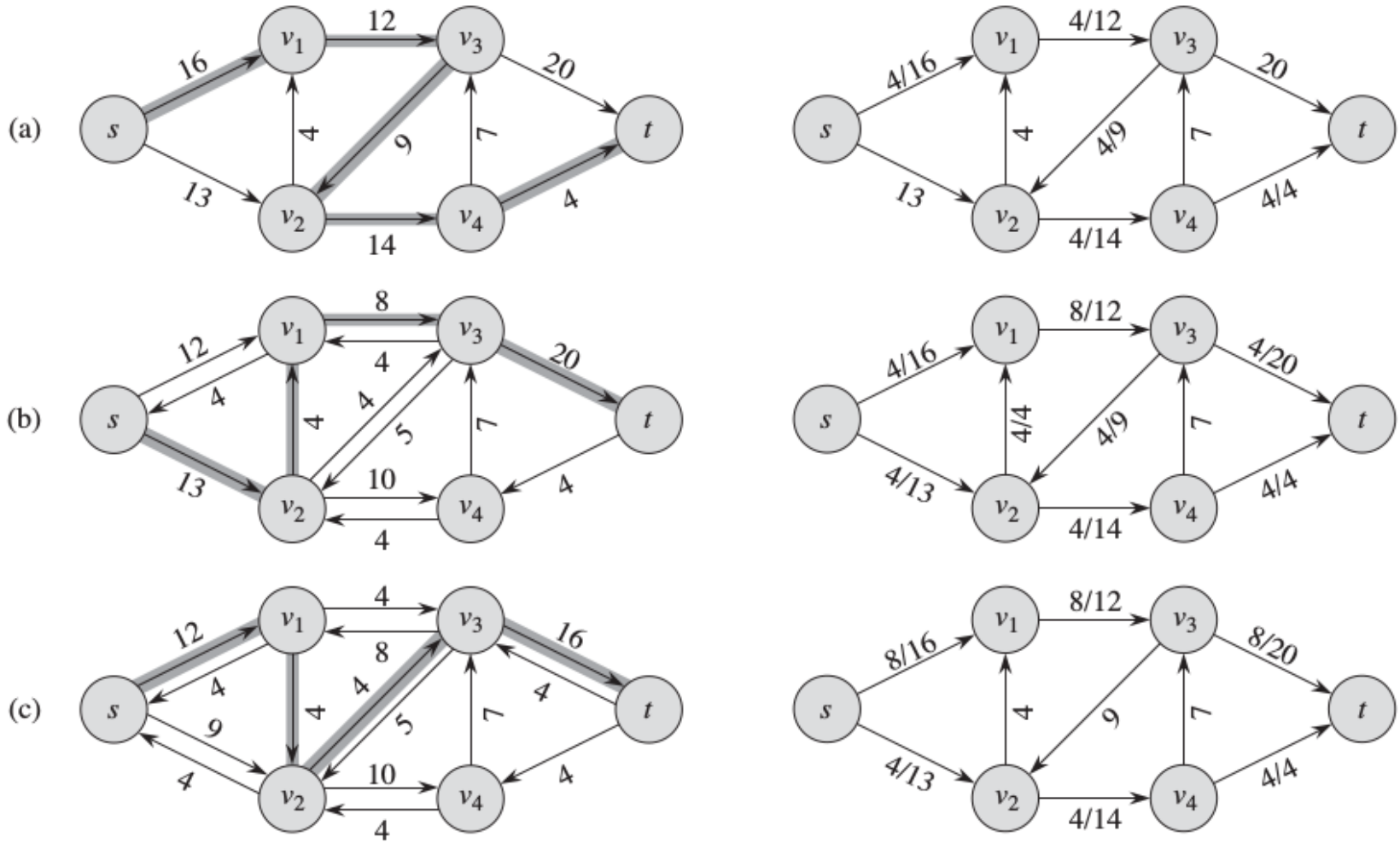
Algorithm 7: Ford-Fulkerson-Method(G,s,t)

```
1 Initialize f to 0
2 while there exists an augmenting path p in the residual network G_f
  do
3     augment flow f along p
4 return f
```

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{otherwise} \end{cases}$$

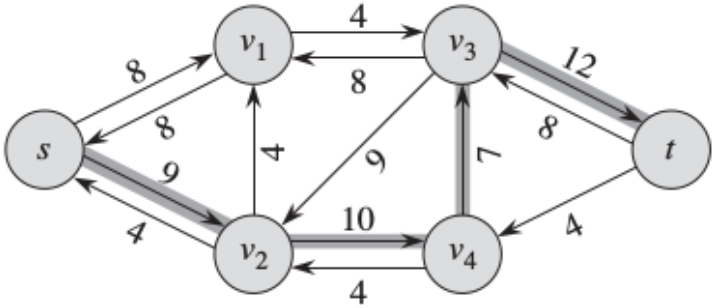
Flow is maximum if and only if its residual network contains no augmenting path.

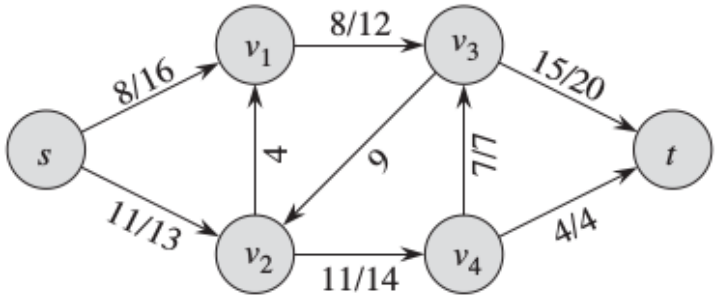
Ford-Fulkerson in Action



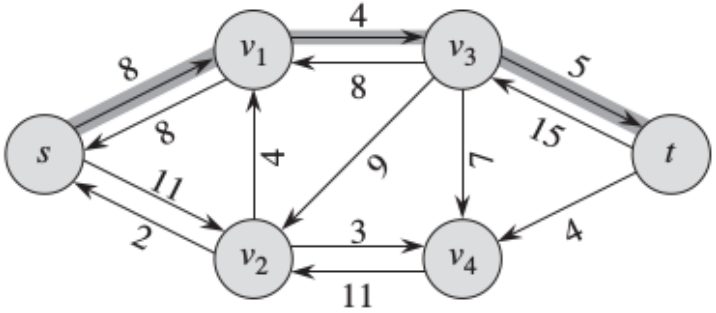
Ford-Fulkerson in Action

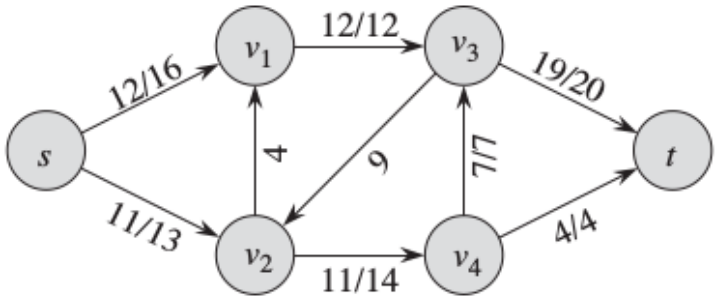
(d)



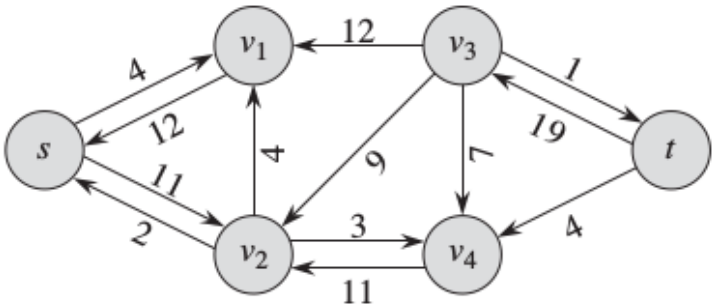


(e)





(f)



Analysis

- In case of rational weights one can scale all the weights to make them integer
- In every iteration flow from s to v increases. Let f^* be maximum flow than it take at most $|f^*|$ iterations to converge
- Every iteration tries to get a path in residual network, it can take $O(E)$ time using DFS or BFS
- Therefore, total running time is $O(|E| \times |f^*|)$

Edmonds-Karp algorithm

Take augmenting path as the shortest path from s to t in residual network

- Total number of flow augmentations performed by the Edmonds-Karp algorithm is $O(|V| \times |E|)$
- Therefore, it takes $O(|E| \times (|V| \times |E|)) = O(|V| \times |E|^2)$ time to find maximum flow.

Maximum Bipartite matching

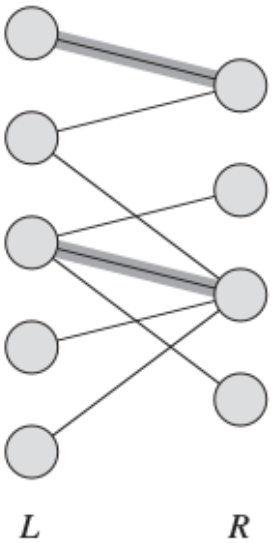
- Bipartite Graph

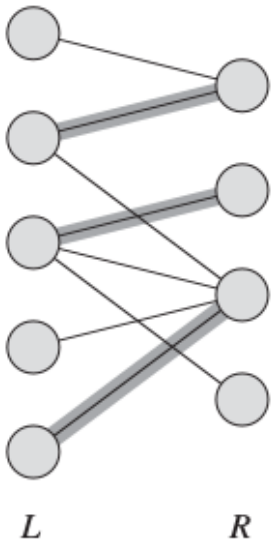
Maximum Bipartite matching

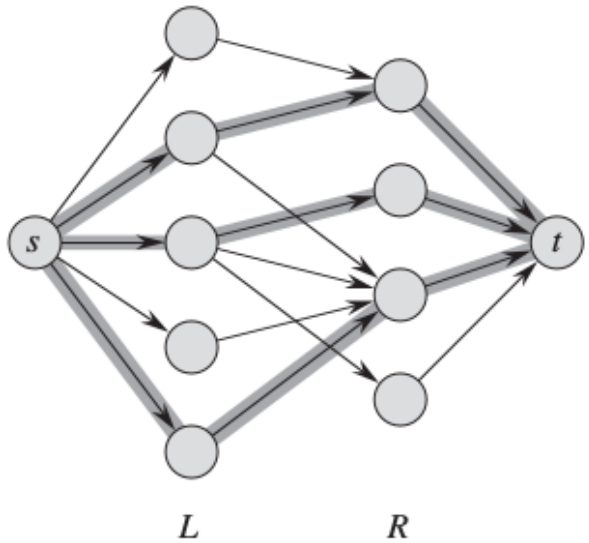
- Bipartite Graph
- Matching in bipartite graph

Maximum Bipartite matching

- Bipartite Graph
- Matching in bipartite graph







Thank You!

Thank you very much for your attention! (Reference[1])

Queries ?

[1] [1] Book - *Introduction to Algorithm*, By THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVEST, CLIFFORD STEIN