# Advanced Algorithms and Parallel Programming Spring 2018

## Advanced Algorithms Project

Semsi Yigit Ozgumus

Politecnico di Milano

July 17, 2018

# Table of Contents

1. Introduction of the Problem
2. Implementation of the Project
3. Analysis Results

# Current Section

1. Introduction of the Problem
2. Implementation of the Project
3. Analysis Results

# What is a Strongly Connected Component ?

# What is a Strongly Connected Component ?

- if a Vertex is reachable by any vertex in the graph, then that vertex is **strongly connected**.

# What is a Strongly Connected Component ?

- if a Vertex is reachable by any vertex in the graph, then that vertex is **strongly connected**.
- if a partition of vertices that are a subgraph of the graph are all strongly connected, then that subgraph is a **strongly connected component** of that graph.

# What is a Strongly Connected Component ?

- if a Vertex is reachable by any vertex in the graph, then that vertex is **strongly connected**.
- if a partition of vertices that are a subgraph of the graph are all strongly connected, then that subgraph is a **strongly connected component** of that graph.

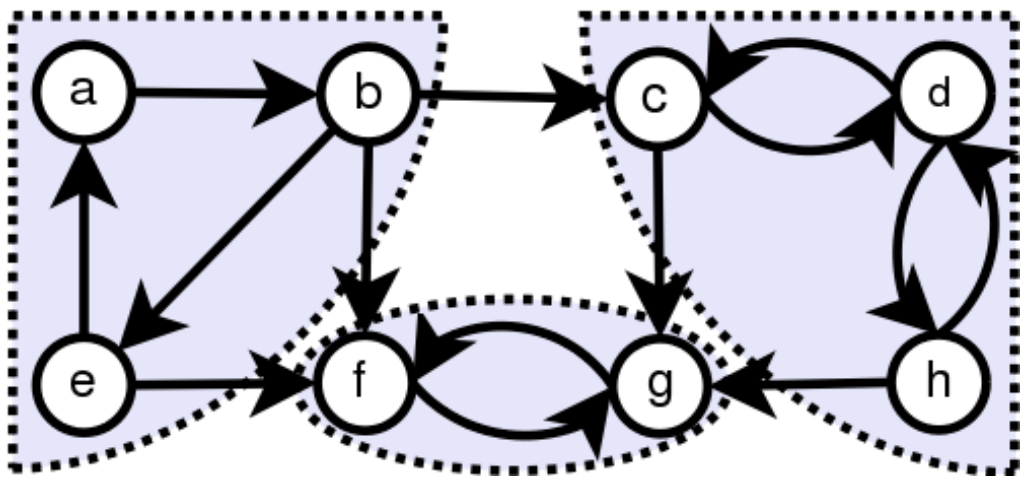


Figure: Graph with strongly connected components

# Methods to solve

# Methods to solve

- Tarjan's Algorithm

# Methods to solve

- Tarjan's Algorithm
  - Time Complexity $\rightarrow O(|V| + |E|)$

# Methods to solve

- Tarjan's Algorithm
  - Time Complexity $\rightarrow O(|V| + |E|)$
  - Space Complexity $\rightarrow O(|V| \cdot (2 + 5w))$

# Methods to solve

- Tarjan's Algorithm
  - Time Complexity $\rightarrow O(|V| + |E|)$
  - Space Complexity $\rightarrow O(|V| \cdot (2 + 5w))$
- Nuutila's Version

# Methods to solve

- Tarjan's Algorithm
  - Time Complexity $\rightarrow O(|V| + |E|)$
  - Space Complexity $\rightarrow O(|V| \cdot (2 + 5w))$
- Nuutila's Version
  - Space Complexity $\rightarrow O(|V| \cdot (1 + 4w))$

# Methods to solve

- Tarjan's Algorithm
  - Time Complexity $\rightarrow O(|V| + |E|)$
  - Space Complexity $\rightarrow O(|V| \cdot (2 + 5w))$
- Nuutila's Version
  - Space Complexity $\rightarrow O(|V| \cdot (1 + 4w))$
- Pearce's Version

# Methods to solve

- Tarjan's Algorithm
  - Time Complexity $\rightarrow O(|V| + |E|)$
  - Space Complexity $\rightarrow O(|V| \cdot (2 + 5w))$
- Nuutila's Version
  - Space Complexity $\rightarrow O(|V| \cdot (1 + 4w))$
- Pearce's Version
  - Space Complexity $\rightarrow O(|V| \cdot (1 + 3w))$

# Current Section

1 Introduction of the Problem

2 Implementation of the Project

3 Analysis Results

# Project Structure

- In terms of functionality the project has 4 different functions.

# Project Structure

- In terms of functionality the project has 4 different functions.
  1. Generate a Random Directed Graph with options (Python)

# Project Structure

- In terms of functionality the project has 4 different functions.
  1. Generate a Random Directed Graph with options (Python)
  2. Run Experiement with algoritmhs while keeping track of completion time and storage consumption (tarjan vs. Nuutila vs. Pearce)

# Project Structure

- In terms of functionality the project has 4 different functions.
  1. Generate a Random Directed Graph with options (Python)
  2. Run Experiement with algoritmhs while keeping track of completion time and storage consumption (tarjan vs. Nuutila vs. Pearce)
  3. Debug mode for testing a singular graph with algorithms provided.

# Project Structure

- In terms of functionality the project has 4 different functions.
  1. Generate a Random Directed Graph with options (Python)
  2. Run Experiement with algoritmhs while keeping track of completion time and storage consumption (tarjan vs. Nuutila vs. Pearce)
  3. Debug mode for testing a singular graph with algorithms provided.
  4. Visualization of the experiments, can only be done with cvs files (Python).

# Project Structure

- All the steps are connected to each other and glued together to create a pipeline to emulate terminal application.

# Project Structure

- All the steps are connected to each other and glued together to create a pipeline to emulate terminal application.
- All documentation done in Doxygen

# Project Structure

- All the steps are connected to each other and glued together to create a pipeline to emulate terminal application.
- All documentation done in Doxygen
- Visualization of the experiment is done with python, csv files can be further explored using the provided jupyter notebook.

# Project Structure

- Application Class uses:

# Project Structure

- Application Class uses:
  - Analyzer, Visualize and Session Classes

# Project Structure

- Application Class uses:
  - Analyzer, Visualize and Session Classes
- Analyzer Class uses:

# Project Structure

- Application Class uses:
  - Analyzer, Visualize and Session Classes
- Analyzer Class uses:
  - Nuutila, Pearce, Tarjan, Timer and GraphComponent Classes

# Project Structure

- Application Class uses:
  - Analyzer, Visualize and Session Classes
- Analyzer Class uses:
  - Nuutila, Pearce, Tarjan, Timer and GraphComponent Classes
- StorageItems struct is used by all algorithm classes to transfer the execution data

# Project Structure

- Application Class uses:
  - Analyzer, Visualize and Session Classes
- Analyzer Class uses:
  - Nuutila, Pearce, Tarjan, Timer and GraphComponent Classes
- StorageItems struct is used by all algorithm classes to transfer the execution data

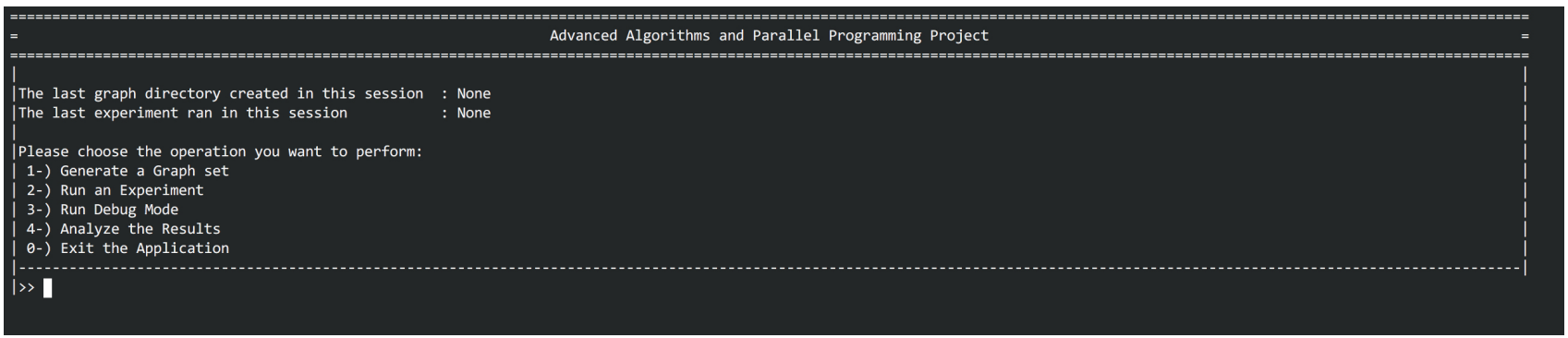


Figure: The main screen of the Application

# Current Section

1 Introduction of the Problem

2 Implementation of the Project

3 Analysis Results

# title

test