



國立臺灣大學  
National Taiwan University

# Recitation Week 14

## PA3 Explanation & Hint

EE4033 Algorithms, Fall 2022

Instructor: James Chien-Mo Li, Yao-Wen Chang, Shao-Hua Sun

Presenter: Shiu-Yun Ding

2022/12/5



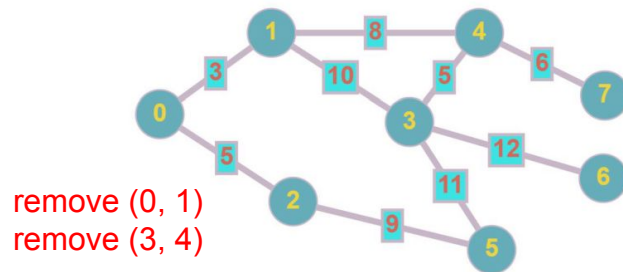
# Outline

---

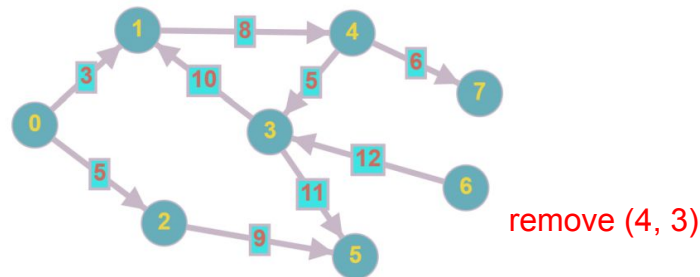
- Problem Formulation
- I/O Format
- Important Assumptions
- Command Lines : PA3 / **Checker** / Submission Checker
- Evaluation
- Submission
- Tips
- Q & A

# Problem Formulation: Cycle Breaking

- Given
  - A graph  $G = (V, E)$  which may contain cycles
  - $V$ : vertex
  - $E$ : edges with weights
- Objective
  - Remove some edges to make the graph **acyclic** and connected with minimum total cost (weight)
  - Cycle Breaking Problem / Cycle Removal Problem



(a) weighted undirected graph



(b) weighted directed graph

# Problem Formulation: Three Types of Graph Instances

---

- Unweighted undirected graph - find optimal solution
  - All edges weights equal to 1
- Weighted undirected graph - find optimal solution
  - General case
  - Including positive/negative/zero edge weights
- Weighted directed graph - no need to find optimal solution
  - Edges are directional
  - Minimum feedback arc set problem
  - NP-hard problem

# Input Format

- 1st line
  - 'u' : undirected graph
  - 'd' : directed graph
- 2nd line
  - an integer  $n$  of the total number of vertices
  - the indice of vertices are from  $0$  to  $n - 1$
- 3rd line
  - an integer  $m$  of the total number of edges
- The following  $m$  lines
  - three integers  $i, j, w$
  - an edge from *vertex  $i$*  to *vertex  $j$*  with *weight  $w$*
  - $w : -100 \sim 100$
- A single 0 at the end of input

Sample Input 1	Sample Input 2
u	d
8	8
9	9
0 1 3	0 1 3
0 2 5	0 2 5
1 3 10	1 4 8
1 4 8	2 5 9
2 5 9	3 1 10
3 4 5	3 5 11
3 5 11	4 3 5
3 6 12	4 7 6
4 7 6	6 3 12
0	0

# Output Format

- 1st line
  - The total weight of removed edges to make the input graph acyclic

- The following lines
  - A list of these removed edges and their weights
  - The order of  $i$  and  $j$  can be different from the input for **undirected graph**
  - The output edges can be in arbitrary order

- If the input graph has no cycles, you should output a line with single “0” (zero)

Sample Output 1	Sample Output 2
8	5
0 1 3	4 3 5
3 4 5	

# Important Assumptions

---

- The input graph has only one connected component
- The output graph (after removing all reported edges) should remain connected
- For undirected graph instances
  - $1 \leq n \leq 100,000$
  - $1 \leq m \leq 50,000,000$
- For directed graph instances
  - $1 \leq n \leq 5,000$
  - $1 \leq m \leq 10,000$

# Command Line - PA3

---

- The executable binary should be named as cb

- Command format:

`./cb [input_filename] [output_filename]`

- Example:

`./cb public_case_1.in public_case_1.out`



# Command Line - Checker

- To verify your results
  - A binary file that can be executed on Linux systems
  - It checks
    - if the output edges are from the input set
    - if the resulted graph is acyclic and connected

- Usage:

`./pa3_checker [input_filename] [output_filename]`

- Example:

`./pa3_checker public_case_1.in public_case_1.out`

# Command Line - Submission Checker

- Create a directory named `<studentID>_pa3` (e.g. r10943109\_pa3/)
  - `src/<all your source code>`
  - `bin/cb`
  - `doc/report.pdf`
  - `makefile`
  - `README`
- Compress your directory into a *tgz* file named `<studentID>_pa3.tgz`  
`tar --exclude='*puts*' -zcvf <studentID>_pa3.tgz <studentID>_pa3/`
- Usage:  
`bash checkSubmitPA3.sh <studentID>_pa3.tgz`

# Evaluation

- Runtime limit for each case : 60 seconds
- 2 cases for unweighted undirected graph (12%)
  - 6 pts for correctness per case
- 3 cases for weighted undirected graph (18%)
  - 6 pts for correctness per case
- 4 cases for weighted directed graph (40%)
  - 2 pts for correctness and 8 pts for performance per case
- README (10%)
- Report (10%)
- Submission Format (10%)

# Submission

---

- Submit your to **<studentID>\_pa3.tgz** NTU COOL before  
**【1pm, December 28, 2022 (Wed.)】**
- Penalty for late submission: 20% per day
- All submissions will be subject to duplication checking  
**【Do Not Plagiarize】**

# Tips

---

- Key words
  - cycle breaking problem, cycle removal problem
  - breadth-first search, depth-first search
  - minimum spanning tree
- How to deal with difficult optimization problems?
  - Develop efficient heuristics
  - Greedy/local search methods
- For directed graph, perhaps...
  - Treat it as undirected graph first
  - Then optimize the solution

# Q & A

Please email TA Shiuan-Yun Ding at [r10943109@ntu.edu.tw](mailto:r10943109@ntu.edu.tw) for any questions

