National University of Singapore

School of Computing

CS2040S - Data Structures and Algorithms

Final Assessment

(Semester 1 AY2023/24)

Time Allowed: 2 hours

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains TWO (2) sections.
   It comprises SIXTEEN (16) printed pages, including this page.
3. This is an **Open Book Assessment**.
4. Answer **ALL** questions within the **boxed space** of the answer sheet (page 13-16).
   For Section A, shade the option in the answer sheet (use 2B pencil).
   There are a few starred (*) boxes: free 1 mark if left blank but 0 for wrong answer (no partial).
   The answer sheet is at page 13-16 but you will still need to hand over the entire paper.
   You can use either pen or pencil. Just make sure that you write **legibly**!
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
   Read all the questions first! Some (subtask) questions might be easier than they appear.
6. You can use **pseudo-code** in your answer but beware of penalty marks for **ambiguous answer**.
   You can use **standard, non-modified** classic algorithm in your answer by just mentioning its name, e.g. run Dijkstra's on graph G, Kruskal's on graph G',
7. All the best :)

A MCQs ($25 \times 2 = 50$ marks)

Select the **best unique** answer for each question.
Each correct answer worth 2 marks.

The MCQ section will not be archived to open up possibilities of reuse in the future.

[PAGE 3 IS NOT ARCHIVED]

[PAGE 4 IS NOT ARCHIVED]

[PAGE 5 IS NOT ARCHIVED]

[PAGE 6 IS NOT ARCHIVED]

[PAGE 7 IS NOT ARCHIVED]

[PAGE 8 IS NOT ARCHIVED]

[Top part of PAGE 9 IS NOT ARCHIVED]

B Applications (20+15+15 = 50 marks)

B.1 Special Graph Traversal (20 marks)

You are given a special unweighted undirected graph that is constructed using the following rule. Let N be a parameter of the graph ($1 \le N \le 10^6$). There are $1 + 3 \times N$ vertices in this graph, numbered from $[0 \ldots 3 \times N]$. There are edges only between the following vertices $(i, i+1)$, $(i, i+2)$, $(i+1, i+3)$, $(i+2, i+3)$, $\forall i \in [0, 3, 6, \ldots, 3 \times (N-1)]$.

An example of this special graph with $N = 2$ is as below:

```
N = 2
0--1
|  |
2--3--4
   |  |
   5--6
```

B.1.1 Draw $N = 4$ (3 marks)

Show that you have understand how to construct this special graph by drawing the graph when $N = 4$. Any valid graph drawing will be accepted.

B.1.2 Store the Graph? (2 marks)

Propose how are you going to store this special graph to facilitate the sub-questions below?

B.1.3 Who Are My Neighbors? (5 marks)

How do you list down the neighbors of a specific vertex $u \in [0 \dots 3 \times N]$ in ascending order? For example, if $N = 2$ and $u = 3$, then the neighbors of $u = 3$ are $\{1, 2, 4, 5\}$. Note that your answer must be consistent with your previous answer in Section B.1.2.

B.1.4 Shortest Path (5 marks)

Design an algorithm to report the *number of edges* along the shortest path from a source vertex s to a destination vertex t in *this special graph.* Analyze the time complexity of your algorithm. You will be graded first by correctness then by efficiency of your algorithm.

B.1.5 With Some Missing Edges (5 marks)

On top of the special graph structure outlined at Section B.1, you are also told that up to k ($0 \le k \le min(100, 4 \times N)$) edges are missing from the special graph. Will your answer for Section B.1.2, B.1.3, and/or B.1.4 change? If you say 'no change', argue why these missing edges will not matter. If you say 'I will modify my answer', outline your modifications.

B.2 Reverse Kruskal's Algorithm (15 marks)

In class, we have learned that we can find the MST of a connected undirected weighted graph $G = (V, E)$ with n vertices ($1 \le n \le 100\,000$) and m edges ($n-1 \le m \le 500\,000$) using Kruskal's algorithm as follows: from an empty tree T, consider the edges $e_0, e_1, \dots, e_{m-1}$ in non-decreasing weight (same or increasing) order. We add edge e_i into T if and only if the two endpoints of $e_i = (u, v)$ are not already connected by a path in T.

B.2.1 Details of Kruskal's (3 marks)

Elaborate the details of Kruskal's on these three sub-points (1 mark each).

1. How to easily sort the set E (of m edges) and what is the time complexity?
2. How to easily check if the two endpoints of $e_i = (u, v)$ are not already connected by a path in T in $O(1)$ without using $O(n + m)$ DFS/BFS?
3. What is the overall time complexity of Kruskal's algorithm?

Alternative MST Algorithm: Reverse Kruskal's

Suppose you are given an alternative MST algorithm that sounds like reverse Kruskal's algorithm as follows: let T be initially contains the entire set of edges E of G. Then, while there is still some cycle C in T, we remove edge e from T where e has the heaviest weight in C.

Your task is to implement this alternative MST algorithm: Given a connected undirected weighted graph $G = (V, E)$, output all edge(s) that is/are the heaviest edge in some cycle of G. You will delete this/these edge(s) to unveil the MST. You can assume that the edge weights of all m edges are distinct and G is not already an MST at the beginning, so you will need to output at least one heavy edge.

B.2.2 Manual Test Case (2 marks)

Given a connected weighted graph $G = (V, E)$ as shown in Figure 1, one of the edge that is the heaviest edge of a cycle is edge $(1, 2)$ with weight 95. It is the heaviest edge of cycle $(0 \to 1 \to 2 \to 0)$, $(1 \to 2 \to 3 \to 1)$, and also $(1 \to 2 \to 3 \to 4 \to 1)$. Find two more edges that are also heaviest edge in some cycle of G.

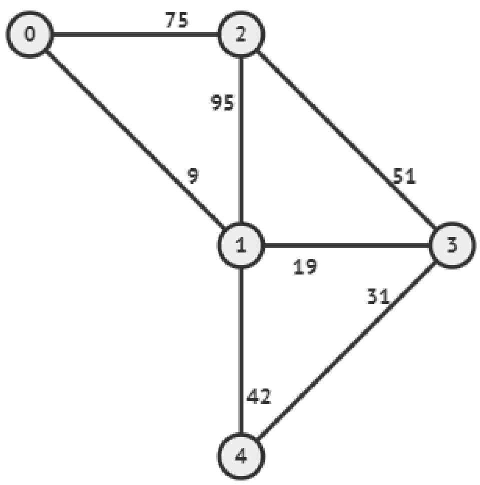


Figure 1: An Example Connected Undirected Weighted Graph $G = (V, E)$

B.2.3 Implement Reverse Kruskal's Algorithm (10* marks)

Now, implement this Reverse Kruskal's algorithm and analyze the time complexity.
You will get 0/1/9/10 for wrong/blank/correct but slow/correct and as fast as normal Kruskal's.

B.3 Alternative Shortest Path (15 marks)

Finding the Single-Source Shortest Path (SSSP) from the source vertex s to a specific destination vertex t given a directed (positive) weighted graph G is a standard CS problem. Since most people who use any good map application will very likely follow these SSSP information, those shortest path(s) - can be plural - will get more and more crowded and thus slower over time.

Now you want to find the Alternative Single-Source Shortest Path (ASSSP) in G, defined as follows: ASSSP is the shortest path of G that goes from vertex s to vertex t such that no edge between two consecutive vertices in ASSSP belongs to *any* SSSP from s to t. Output the weight of the ASSSP.

B.3.1 Manual Test Case (2 marks)

Given a directed (positive) weighted graph $G = (V, E)$ as shown in Figure 2, there are two SSSPaths from $s = 0$ to $t = 6$. They are path X: $0 \to 1 \to 2 \to 6$ with shortest path weight 4 and path Y: $0 \to 4 \to 6$, also with shortest path weight 4. The weight of the ASSSP for this test case is 5 as path $0 \to 3 \to 6$ do not traverse any edge that is used in shortest path X or shortest path Y.

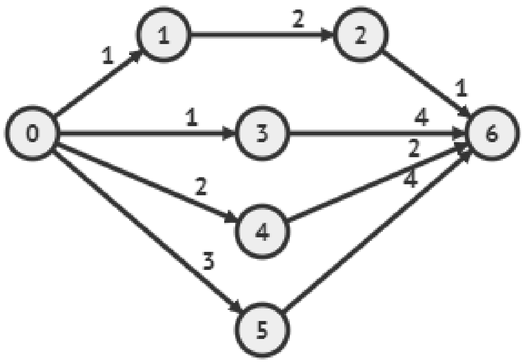


Figure 2: An Example Directed Weighted Graph $G = (V, E)$

0. Extra example: If edge $(3, 2)$ with weight 2 is added into Figure 2 (independent on the two changes below), the weight of ASSSP becomes: 7, and the ASSSP is path: $0 \to 5 \to 6$.

1. If edge $(0, 3)$ in Figure 2 is changed from weight 1 to weight 4 (independent on the change below), the weight of ASSSP becomes: _____, and the ASSSP is path: $0 \to$ __________ $\to 6$.

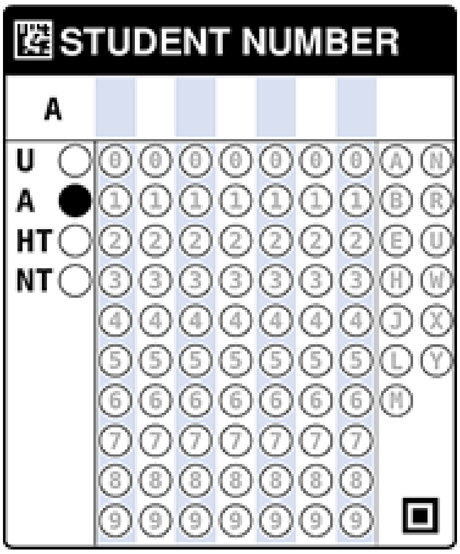
2. If edge $(1, 2)$ in Figure 2 is changed from weight 2 to weight 1 (independent on the change above), the weight of ASSSP becomes: _____, and the ASSSP is path: $0 \to$ __________ $\to 6$.

B.3.2 Solve the ASSSP Problem (13* marks)

Propose how you are going to identify the weight of ASSSP given $G = (V, E)$ with n vertices $(1 \leq n \leq 100\,000)$ and m edges $(n - 1 \leq m \leq 500\,000)$ and analyze your time complexity! Note that if there is no possible ASSSP in G, you have to output -1 instead.

The Answer Sheet

Write your Student Number and MCQ answers in the boxes below using **(2B) pencil**:



No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
B	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
C	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
E	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

No	16	17	18	19	20	21	22	23	24	25
A	○	○	○	○	○	○	○	○	○	○
B	○	○	○	○	○	○	○	○	○	○
C	○	○	○	○	○	○	○	○	○	○
D	○	○	○	○	○	○	○	○	○	○
E	○	○	○	○	○	○	○	○	○	○

Box B.1.1. Draw $N = 4$.

Box B.1.2. Store the Graph?

Box B.1.3. Who Are My Neighbors?

Box B.1.4. Shortest Path

Box B.1.5. With Some Missing Edges

In case this sheet is detached from page 13-14, re-write your Student Number again:

STUDENT NUMBER					
A					

Box B.2.1. Details of Kruskal's (write 3 lines)

Box B.2.2. Two More Edges (write 2 lines)

Box B.2.3.* (1 if blank, 0 if wrong) Implement Reverse Kruskal's Algorithm!

Box B.3.1. Two Manual Test Cases (write 2 lines)

Box B.3.2.* (1 if blank, 0 if wrong) Solve the ASSSP Problem

– END OF PAPER; All the Best –