NATIONAL UNIVERSITY OF SINGAPORE

CS2040S—Data Structures and Algorithms

2021/2022 Semester 2

Time Allowed: 2 hours

<u>INSTRUCTIONS TO STUDENTS</u>

1. Please write your Student Number only. Shade the circles corresponding to your student number correctly. Do not write your name.
2. The assessment paper contains **TWELVE (12) questions** and comprises **TWENTY-FOUR (24) pages** including this cover page.
3. Weightage of each question is given in square brackets. The maximum attainable score is 100.
4. This is a **<u>CLOSED</u>** book assessment, but you are allowed to bring **<u>ONE</u>** double-sided A4 sheet of notes for this assessment. You may not bring any magnification equipment! You may NOT use a calculator, your mobile phone, or any other electronic device.
5. Write all your answers in the space provided in the **ANSWER BOOKLET**.
6. You are allowed to write with pencils, as long as it is legible.
7. Shade the answer circles fully and make sure the shading is dark enough.
8. Read through the problems before starting. Do not spend too much time on any one problem.
9. For the multiple choice questions, no partial credit will be given. There is intended to be only one correct answer per multiple choice question.
10. For the open-ended questions, partial credit may be given, so show your work and explain your assumptions. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it.

This page is intentionally left blank.

It may be used as scratch paper.

Question 1: True or False [7 marks]

For each of the following questions in this section, specify whether the statement is true or false by shading in the proper bubble on the answer sheet.

A. Binary search works correctly on an array with repeated values, as long as the values in the array are non-decreasing (i.e., for all $i < n, A[i] \leq A[i+1]$). [1 mark]

B. Function $f(x)$ is non-decreasing on the input values $[1 \ldots n]$. Function $g(x)$ is non-increasing on the input values $[1 \ldots n]$. You can use the peak finding algorithm to find the maximum value of $f + g$ on the input values $[1 \ldots n]$. [1 mark]

C. $T(n) = 2T(n/2) + n^2$ and $T(1) = 1$. Then $T(n) = \Theta(n^2 \log n)$. [1 mark]

D. Assume you have a sorted array containing n items. Ten arbitrary items from the array are removed and replaced with ten new items. Then running InsertionSort on the resulting array will take $O(n)$ time. [1 mark]

E. Halle Ashtable has invented a new type of symbol table where every insert and search operation takes amortized $O(1)$ time. Her boss Frank Frazzle is very excited: he believes that with this new data structure, if you start with an empty table and execute any sequence of k insert operations and ℓ search operations, in any order, then the execution will complete in $O(k + \ell)$ time, in the worst case. Assuming Halle's data structure works as claimed, then is Frank correct? [1 mark]

F. In the inner loop of Bellman-Ford, each edge in the graph is relaxed exactly once. In a connected, directed, weighted graph with no negative weight cycles, after j iterations of the inner loop of Bellman-Ford, every node within j hops of the source has an estimate equal to its correct distance from the source. [1 mark]

G. Assume you have a connected, weighted graph G with no positive weight cycles. Then you can efficiently find the longest path in the graph by negating all the edge weights and running Dijkstra's algorithm. [1 mark]

Question 2: Explain Why [6 marks]

For each of the following questions in this section, we compare an algorithm implementation choice with an alternative. Your goal is to determine why we made that choice. Was it because it provides better performance? Was it because it results in a more correct algorithm? Is either choice equally good? Or is the choice specified wrong, i.e., a good implementation of the algorithm does not make the specifid choice?

A. Use BFS instead of DFS on the maze assignment to find the shortest escape route.
[1 mark]

B. Use an adjacency matrix representation instead of an adjacency list representation for sparse graphs (e.g., when finding shortest paths). [1 mark]

C. Sort the edges by weight in the first step of Kruskal's, instead of leaving them in an arbitrary order. [1 mark]

D. Use BFS instead of Bellman-Ford to find shortest paths in a graph where all the edge weights are identical. [1 mark]

E. Use post-order DFS instead of Kahn's Algorithm to find a topological ordering.
[1 mark]

F. Use priority queue instead of a queue in Prim's Algorithm. [1 mark]

Question 3: Shortest Paths [5 marks]

For each of the following properties, specify which of the shortest path algorithms that we have studied has that property. (Assume in all cases that the algorithms are running on connected, weighted graphs with positive edge weights, unless otherwise specified.)

A. Each node v stores an estimate `est[v]`. During the execution (after initialization), if `est[v]` is equal to an integer, then it is equal to the distance from s to v along some path.
[1 mark]

B. Each edge is relaxed at most once. [1 mark]

C. Each node v stores an estimate `est[v]`. During the execution (after initialization), `est[v] >=` the shortest path distance from s to v. [1 mark]

D. Each node v stores an estimate `est[v]`. During the execution (after initialization), `est[v] <=` the shortest path distance from s to v. [1 mark]

E. Each node v stores an estimate `est[v]`. Let d be the target destination, and let node u be a node on the shortest path from s to d. The first time that we relax an edge (u,d) and the estimate at d is reduced, it would be safe to terminate the algorithm (since we have already found a shortest path from s to d). [1 mark]

Question 4: DAG Breadth First Search? [11 marks]

Below is pseudocode for a (possibly incorrect) breadth-first-search algorithm designed specifically for directed acyclic graphs (DAGs). It takes as input a set of nodes V, a set of edges E, and a source node s. Assume that Queue is a typical implementation of a standard queue supporting three methods: `enqueue`, `dequeue`, and `isEmpty`. For a node v, the function `nbrs(v)` returns a set of neighbos of v. It returns an array `est` containing the distance from the source to each node in the graph. Assume for the remainder of the problem that the graph $G = (V, E)$ is an unweighted, connected directed acyclic graph.

```
DAG_BFS(V, E, s)
n = V.size()  // number of nodes
est = new int[n]
queue = new Queue(n)
queue.enqueue(s)
while !queue.isEmpty():
     v = queue.dequeue()
     for each (w in nbrs(v)):
          est[w] = est[v] + 1
          queue.enqueue(w)
return est
```

A. The `DAG_BFS` algorithm above terminates on every connected, unweighted directed acyclic graph. True or false? [2 marks]

B. Each node is removed from the queue at most once. True or false? [1 mark]

C. The queue may contain the same node more than once at the same time. True or false? [1 mark]

D. If a node has k neighbors, than each node (aside from the source s) is added to the queue at most k times. (Post-exam correction: better wording would be: if a node has k neighbors, then it is added to the queue at most k times. However, either interpretation yields the same answer.) True or false? [1 mark]

E. Assume that G is a DAG for which the algorithm does terminate. Then when the algorithm terminates, `est[v]` equals which of the following?

A. The length of the (unweighted) shortest path from the source to v.

B. The length of the (unweighted) longest path from the source to v.

C. The number of distinct paths from the source to v.

D. None of the above.

[3 marks]

F. Assume that G is a DAG for which the algorithm does terminate. Then what is the tightest asymptotic bound (of those listed below) on the running time of the `DAG_BFS` algorithm? Let n be the number of nodes in the graph and m the number of edges in the graph.

A. $O(n)$

B. $O(n+m)$

C. $O(nm)$

D. $O(nm^2)$

E. $O(2^{2m})$

F. None of the above.

[3 marks]

Question 5: Dijkstra's Algorithm [9 marks]

In this question, you will complete an implementation of Dijkstra's Algorithm by filling in the blanks. (Notice that the code given is pseudo-Java code, but may not compile as Java.) The function takes as input a graph G and a source node s and returns an array consisting of the shortest distance from s to each node.

Each node is represented by an integer. If there are n nodes in the graph, the nodes are the set $\{0,\ldots,n-1\}$. (Note that the function takes `int s` to indicate the source node.) The `Graph` class supports a few useful functions:

- `G.size()` returns the number of nodes in the graph.
- `G.adjacent(int u)` returns a collection of nodes adjacent to node `u`. (This collection can be iterated through via a for loop.)
- `G.weight(int u, int v)` returns the weight of the edge (u,v) if it is in the graph, or `POSITIVE_INFINITY` otherwise.

Thoughout, we will use `NEGATIVE_INFINITY` as a constant smaller than any integer that can be used to represent negative infinity; we will use `POSITIVE_INFINITY` as a constant larger than any integer that can be used to represent positive infinity. (Otherwise, they can be manipulated just like integers.) All edge weights are positive integers.

The algorithm uses various other data structures, e.g., stacks, queues, arrays, priority queues, etc. You should assume that each of these is implemented in a reasonable and efficient way for the specified abstract data type.

There are, of course, many different ways to implement Dijkstra's Algorithm. You should choose a manner to fill in the blanks that results in a correct and efficient implementation of Dijkstra's Algorithm.

```
int[] Dijkstra(Graph G, int s)
{
    int n = G.size(); // number of nodes in graph

    int est[] = new int[n]; // used to store estimates
    for (int v=0; v<n; v++) {
        est[v] = (Q5A:______);
    }

    est[s] = (Q5B:______);

    // Create data structure ds to keep track of nodes.
    (Q5C:_______)

    // Add all nodes initially to the data structure.
    for (int v=0; v<n; v++) {
        (Q5D:_______);
    }

    while (!ds.isEmpty()) {
        int u = (Q5E:______)

        for (int v : G.adjacent(u)) {
            if ((Q5F:______) {
                // Update estimate of v
                est[v] = (Q5G:______);
                // Update data structure ds
                (Q5H:_____)
            }
        }

    }

    // Return the shortest path distances from s
    return est;
}
```

A. Fill in the blank from the following options:

A. `POSITIVE_INFINITY`

B. `NEGATIVE_INFINITY`

C. `0`

D. `1`

E. `-1`

[1 mark]

B. Fill in the blank from the following options:

A. `POSITIVE_INFINITY`

B. `NEGATIVE_INFINITY`

C. `0`

D. `1`

E. `-1`

[1 mark]

C. Fill in the blank from the following options:

A. `PriorityQueue ds = new PriorityQueue();`

B. `Queue ds = new Queue();`

C. `Stack ds = new Stack();`

D. `int ds[] = new int[n];`

[1 mark]

D. Fill in the blank from the following options:

A. `ds.insert(v, est[v])`

B. `ds.increaseKey(v, est[v])`

C. `ds.decreaseKey(v, est[v])`

D. `ds.extractMin()`

E. `ds.extractMax()`

F. `ds.enqueue(v)`

G. `ds.dequeue()`

H. `ds.push(v)`

I. `ds.pop()`

J. `ds[0]`

K. `ds[v] = est[v]`

L. `ds[s]`

[1 mark]

E. Fill in the blank from the following options:

A. `ds.insert(v, est[v])`
B. `ds.increaseKey(v, est[v])`
C. `ds.decreaseKey(v, est[v])`
D. `ds.extractMin()`
E. `ds.extractMax()`
F. `ds.enqueue(v)`
G. `ds.dequeue()`
H. `ds.push(v)`
I. `ds.pop()`
J. `ds[0]`
K. `ds[v] = est[v]`
L. `ds[s]`

[1 mark]

F. Fill in the blank from the following options:

A. `est[u] < est[v] + G.weight(u,v)`
B. `est[v] < est[u] + G.weight(u,v)`
C. `est[v] > est[u] + G.weight(u,v)`
D. `est[u] > est[v] + G.weight(u,v)`
E. `est[u] == est[v] + G.weight(u,v)`
F. `est[u] + G.weight(u,v)`
G. `est[v] + G.weight(u,v)`
H. `G.weight(u,v)`

[1 mark]

G. Fill in the blank from the following options:

A. `est[u] < est[v] + G.weight(u,v)`
B. `est[v] < est[u] + G.weight(u,v)`
C. `est[v] > est[u] + G.weight(u,v)`
D. `est[u] > est[v] + G.weight(u,v)`
E. `est[u] == est[v] + G.weight(u,v)`
F. `est[u] + G.weight(u,v)`
G. `est[v] + G.weight(u,v)`
H. `G.weight(u,v)`

[1 mark]

H. Fill in the blank from the following options:

A. `ds.insert(v, est[v])`
B. `ds.increaseKey(v, est[v])`
C. `ds.decreaseKey(v, est[v])`
D. `ds.extractMin()`
E. `ds.extractMax()`
F. `ds.enqueue(v)`
G. `ds.dequeue()`
H. `ds.push(v)`
I. `ds.pop()`
J. `ds[0]`
K. `ds[v] = est[v]`
L. `ds[s]`

[1 mark]

I. We would like to modify this implementation of Dijkstra to find the shortest path tree (and not just the distances). We want to compute, for each node `u`, the `parent[u]`, i.e., the parent of node u in the shortest path tree. Assume that we create an integer array `parent[]` of size `n` and initialize each entry to -1. While of the following additions will correctly implement this?

A. Add the line `parent[u] = v` immediately after line 24.

B. Add the line `parent[u] = v` immediately after line 29.

C. Add the line `parent[v] = u` immediately after line 24.

D. Add the line `parent[v] = u` immediately after line 29.

E. None of the above options will yield a correct shortest path tree.

[1 mark]

Question 6: How fast is it? [6 marks]

A. Let G be a connected, directed, weighted graph stored as an adjacency matrix. Graph G has n nodes and m edges. What is the (asymptotic) worst-case running time of Breadth-First-Search on G? (Give the tightest possible asymptotic bound.)

A. $O(n)$

B. $O(n+m)$

C. $O(m\log n)$

D. $O(n^2)$

E. $O(mn)$

F. $O(n^3)$

G. $O(mn^2)$

[2 marks]

B. Let G be a connected, directed, weighted graph stored as an adjacency matrix. Graph G has n nodes and m edges. What is the (asymptotic) worst-case running time of Prim's Algorithm on G? Assume Prim's uses a priority queue implemented with an AVL tree. (Give the tightest possible asymptotic bound.)

A. $O(n)$

B. $O(n+m)$

C. $O(m\log n)$

D. $O(n^2)$

E. $O(mn)$

F. $O(n^3)$

G. $O(mn^2)$

[2 marks]

C. Assume you are given a special priority queue `SuperQueue` that implements the following operations with the following costs, assuming there are n elements in the priority queue:

- `insert`: $\Theta(\log^2 n)$
- `extractMin`: $\Theta(n)$
- `decreaseKey`: $\Theta(\log n)$
- `isEmpty`: $\Theta(1)$

What is the running time for Dijkstra's Algorithm on a connected, directed, weighted graph with n nodes and m edges if it uses the `SuperQueue` priority queue? Assume the graph is stored as an adjacency list, and all the edge weights are positive. (Give the tightest possible asymptotic bound.)

A. $O(m+n)$
B. $O(n\log^2 n + m\log n)$
C. $O(m\log n)$
D. $O(n^2)$
E. $O(n^2 + m\log n)$
F. $O(n^2\log n + m\log n)$
G. $O(m\log^2 n + n\log n)$
H. $O(mn)$
I. $O(mn^2)$

[2 marks]

Question 7: Which algorithm do you use? [10 marks]

For each of the following problems, choose which algorithm is most appropriate for solving it. Choose the simplest, most efficient algorithm that will find the correct answer. You can use the specified algorithm one or more than one times, as needed. (E.g., if you need to run Prim's Algorithm on several different graphs to solve the problem, that is still a valid use of Prim's; however, you need to take into account the total cost of all the executions.)

A. You are planning an underground subway system designed to efficiently connect the university to twelve different important locations in the city. Digging tunnels is expensive, so your primary goal is to minimize the total tunnel length that needs to be dug. Which algorithm should you use to determine the routes connecting the stations?

A. Breadth-First Search (BFS)
B. Bellman-Ford
C. Dijkstra's Algorithm
D. Prim's Algorithm
E. DAG-shortest-paths
F. Floyd-Warshall
G. Travelling Salesman Approximation (from Problem Set 8)
H. Cannot be solved efficiently.

[2 marks]

B. You want to solve an interesting slider puzzle called "Spin Out" in the fewest number of moves. The puzzle consists of 7 rotating pieces each of which can be in a fixed number of positions. (For this problem, a running time that is exponential in the number of rotating pieces is considered sufficiently efficient.)

A. Breadth-First Search (BFS)
B. Bellman-Ford
C. Dijkstra's Algorithm
D. Prim's Algorithm
E. DAG-shortest-paths
F. Floyd-Warshall
G. Travelling Salesman Approximation (from Problem Set 8)

[2 marks]

C. You are a taxi driver and want to find which routes pay the most. For each possible start and end point of a trip, you take a passenger along the cheapest route. You have a map consisting of road segments connecting intersections, and for each road segment of the city, you know how much you are paid. (You are paid more for certain roads, e.g., near Marina Bay, and less for others.) Find the start and end location that will yield the highest fare for transporting a passenger between those locations. Assume that the road network is sparse.

A. Breadth-First Search (BFS)
B. Bellman-Ford
C. Dijkstra's Algorithm
D. Prim's Algorithm
E. DAG-shortest-paths
F. Floyd-Warshall
G. Travelling Salesman Approximation (from Problem Set 8)
H. Cannot be solved efficiently.

[2 marks]

D. You run a shipping company (Acme Corp) that moves products between n different ports. You can arrange for shipping between any pair of ports. For each pair of ports, you know how much money you will be paid for a given shipment and what the costs of the shipment will be. Your profit is the amount your are paid minus the costs. (For any given pair of ports, the amount you will be paid may be more or less than the costs.) Find a sequence of ports starting in Singapore and returning to Singapore that will make you the most profit.

A. Breadth-First Search (BFS)
B. Bellman-Ford
C. Dijkstra's Algorithm
D. Prim's Algorithm
E. DAG-shortest-paths
F. Floyd-Warshall
G. Travelling Salesman Approximation (from Problem Set 8)
H. Cannot be solved efficiently.

[2 marks]

E. You run another shipping company (Nadir Corp) that competes with Acme Corp; however, you only service routes that make a positive profit. You refuse to arrange shipping for a pair of ports where the profit is not positive. In more detail: you run a shipping company (Nadir Corp) that moves products between n different ports. You can arrange for shipping between any pair of ports. For each pair of ports, you know how much money you will be paid for a given shipment and what the costs of the shipment will be. Your profit is the amount your are paid minus the costs. For any given pair of ports, the profit is positive. Find a sequence of ports starting in Singapore and returning to Singapore that will make you the most profit.

A. Breadth-First Search (BFS)

B. Bellman-Ford

C. Dijkstra's Algorithm

D. Prim's Algorithm

E. DAG-shortest-paths

F. Floyd-Warshall

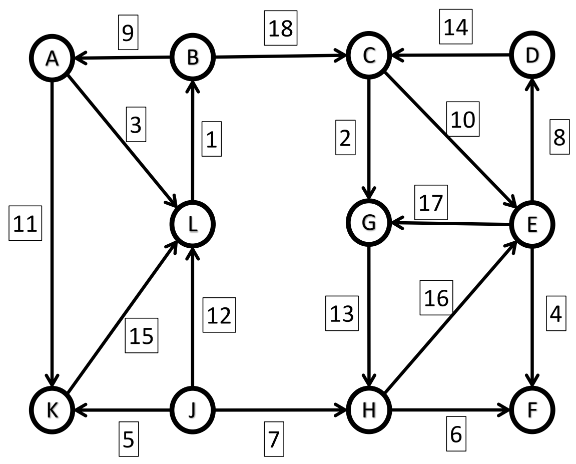
G. Travelling Salesman Approximation (from Problem Set 8)

H. Cannot be solved efficiently.

[2 marks]

Question 8: Some Graph Questions [18 marks]

Consider the following directed, weighted graph with 11 nodes and 18 edges:



A. How many strongly connected components does the graph have?

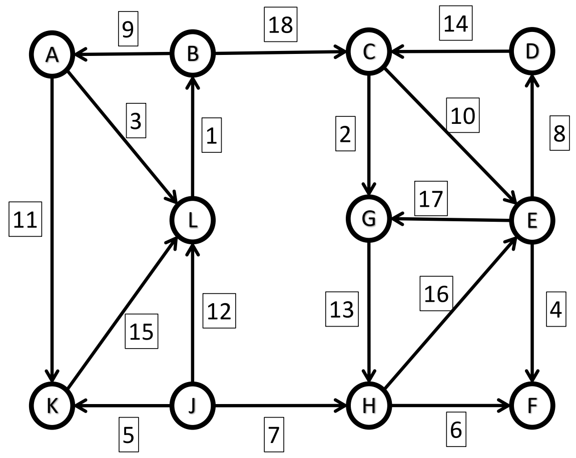
A. 1
B. 2
C. 3
D. 4
E. 5
F. 6
G. 7
H. 8 or more

[2 marks]

B. Does the graph have a unique topological order?

A. It has a unique topological order.
B. It has a topological order, but it is not unique.
C. It does not have a topological order, but if we delete one edge then it will have a topological order.
D. It does not have a topological order, and we need to delete more than one edge for it to have a topological order.

[2 marks]



C. Perform a Depth First Search (DFS) on the graph strating at node A. Assume that the graph is stored as an adjacency list, and each adjacency list is stored in sorted order. Assume A is the first node visited. Which node is the seventh node visited? (Skip repeated nodes when counting. For example, if the traversal began at J, then visited H, then returned to J, then visited K, we would say that K was the third node visited since we are skipping the repeated visit to J.)

A. A
B. B
C. C
D. D
E. E
F. F
G. G
H. H
I. I
J. J
K. K
L. L

[2 marks]

D. Perform a Breadth First Search (BFS) on the graph strating at node A. Assume that the graph is stored as an adjacency list, and each adjacency list is stored in sorted order. Assume A is the first node visited. Which node is the seventh node visited? (Skip repeated nodes when counting. For example, if the traversal began at J, then visited H, then returned to J, then visited K, we would say that K was the third node visited since we are skipping the repeated visit to J.)

A. A
B. B
C. C
D. D
E. E
F. F
G. G
H. H
I. I
J. J
K. K
L. L

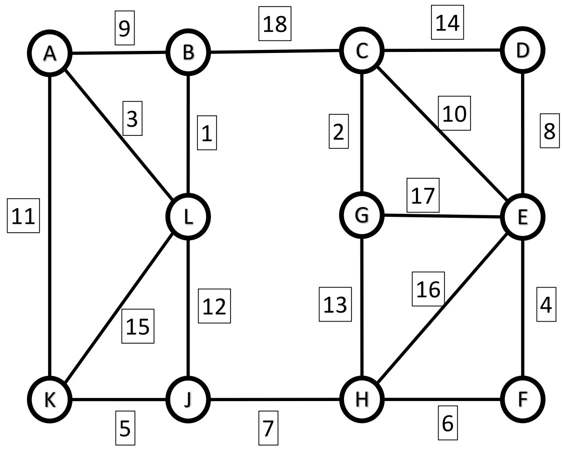
[2 marks]

E. Perform Dijkstra's Algorithm on the graph starting at node A. Assume that the graph is stored as an adjacency list, and each adjacency list is stored in sorted order. Assume A is the first node examined. Which node is the fourth node examined?

A. A
B. B
C. C
D. D
E. E
F. F
G. G
H. H
I. I
J. J
K. K
L. L

[2 marks]

F. Consider the same graph where all the edges are undirected. Run Kruskal's Algorithm on the graph. Which is the first edge examined by the algorithm which is not included in the MST?



A. 1
B. 2
C. 3
D. 4
E. 5
F. 6
G. 7
H. 8
I. 9
J. 10
K. 11
L. 12
M. 13
N. 14
O. Edge 15 or above.

[2 marks]

G. Perform Prim's Algorithm on the graph strating at node A. Assume that the graph is stored as an adjacency list, and each adjacency list is stored in sorted order. Assume A is the first node removed from the priority queue. Which node is the sixth node removed from the priority queue?

A. A
B. B
C. C
D. D
E. E
F. F
G. G
H. H
I. I
J. J
K. K
L. L

[2 marks]

Assume that we have found a minimum spanning tree (MST) for the undirected, weighted, connected graph in the figure above. For each of the statements below, indicate whether it is true or false. (Note that a statement is true only if it correctly identifies whether or not the edge is in the MST and if the because part is a sufficient reason for why the edge is or is not in the MST.)

H. True or false: edge (J,H) is in the MST because it is the lightest edge across a cut. [1 mark]

I. True or false: edge (B,C) is not in the MST because it is the heaviest edge across a cut. [1 mark]

J. True or false: edge (J,L) is not in the MST because it is impossible for every outgoing edge of one node to be in the MST. [1 mark]

K. True or false: edge (H,E) is not in the MST because it is the heaviest edge on the cycle (H,E,C,G,H). [1 mark]

Question 9: Hashing & Hash Tables [10 marks]

This is a problem about hash tables. Assume our hash table is 1-indexed, i.e., the first bucket is 1. There are 13 buckets in our hash table, and the last bucket is number 13. Consider the following hash function that maps elements to a table of size 13:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
11	11	5	2	5	9	1	1	4	4	5	9	13	5	4	11	2	10	7	6	5

Consider the following sequence of items added to the hash table:

G	R	E	A	T	S	P	O	C	K

A. If the hash table resolves collisions via chaining, then how many elements are there in the longest linked list?

A. 1
B. 2
C. 3
D. 4
E. 5
F. 6
G. None of the above

[3 marks]

B. If the hash table resolves collisions via linear probing (i.e., a form of open addressing), then after the previous sequence ("GREATSPOCK") we search for a 'F' in the table. What is the last cell examined before the search returns that there is no 'F' in the table?

A. 1
B. 2
C. 3
D. 4
E. 5
F. 6
G. 7
H. 8
I. 9
J. 10
K. 11
L. 12
M. 13

[3 marks]

C. Which hashing technique guarantees that every insertion will succeed (eventually), regardless of the table load? (Do not consider table resizing.)

[1 mark]

A. Chaining
B. Open addressing

C. Both

D. Neither

D. Which hashing technique guarantees $O(1)$ worst-case searches when the table load $\alpha = 1/2$? [1 mark]

A. Chaining

B. Open addressing

C. Both

D. Neither

E. Which hashing technique guarantees $O(1)$ expected cost searches when the table load $\alpha = 1/2$, as long as the simple uniform hashing assumption or uniform hashing assumption holds (whichever is appropriate)? [1 mark]

A. Chaining

B. Open addressing

C. Both

D. Neither

F. True or false: hashing with chaining (under the Simple Uniform Hashing Assumption) is asymptotically faster (in expectation) than open adressing (under the Uniform Hashing Assumption) when the table contains n elements and has size $m = n + n/\log n$. (There is no table resizing, as we are only interested in operations when the table has the specified size and contains the specified number of elements.)

[1 mark]

Question 10: Algorithm Design: Road Trip [15 marks]

[15 marks]

In this section, please give short and clear answers to each of the questions. If your answers are not clear, you will not get credit—even if your underlying idea is correct! Part of the exam is testing whether you can explain yourself clearly. (We will not accept further explanations after the exam is over.) For example, if we cannot read your handwriting, we will not understand your idea. If you write too much, we may not be able to figure out which part is your answer. If you give several different answers or explanations, we may not be able to determine which one you intend. Please give answers that are short and clear.

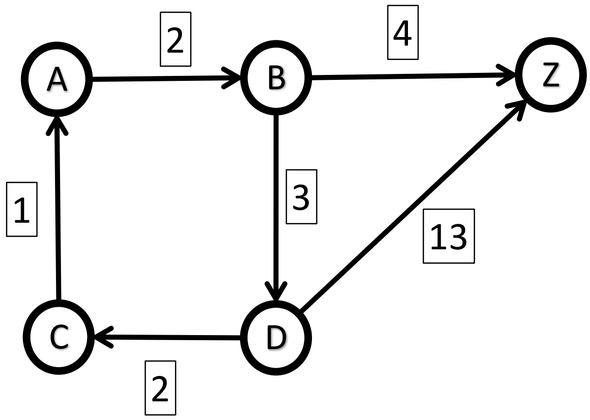
The questions in this section do not have individual marks specified for each subpart. The marks given will depend on the algorithm, its efficiency, and how well it is explained.

Alice, Bob, and Carol are going on a road trip! They are driving from Alberta (Canada) to Zapopan (Mexico). They are going to split up the driving, taking turns behind the wheel, cruising down the open highway to the sounds of the CS2040S Spotify Playlist.

There is one potential problem: Alice, Bob, and Carol are very competitive. If the driving is not exactly equally fair, there will be a fight and the entire trip will be ruined.

At the beginning of the trip, they consult a map. The roads on the map are divided into segments equivalent to one hour of driving. They will alternate driving (Alice, Bob, Carol, Alice, Bob, etc.) each for one hour. They want to find a route to Zapopan that minimizes the distance driven, while guaranteeing that when they arrive each of them will have driven for exactly the same number of hours.

Consider the following example of a map from A(lberta) to Z(apopan):

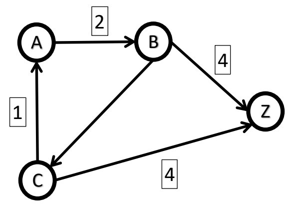


If our three friends drive directly from $(A \to B \to Z)$, the distance is only 6. However, the trip is a failure because Alice and Bob each drive one hour, while Carol drives zero hours. If they drive from $(A \to B \to D \to Z)$, the trip has a distance of 18 and is a success: each drives one hour. There is an even better route, however: from $(A \to B \to D \to C \to A \to B \to Z)$. This route has each of our friends driving for two hours, but the total distance is only 14.

Your goal is to help our friends Alice, Bob, and Carol by designing an algorithm that will help them find the shortest route that ensures they will arrive at Z(apopan) having driven an equivalent number of hours.

A. We want to solve this problem via graph transformation. Explain (briefly, in one or two sentences) how to construct a graph that will make this problem easy to solve. What are the nodes in the new graph? What are the edges in the new graph?

B. Illustrate how to construct the desired graph using the following (simple) example:



Assuming this is the original graph, draw the new graph that you would like to construct.

C. If the map has n nodes and m road segments (each one hour long), how long does it take to construct the new graph? Assume that the map is initially represented as a directed, weighted, connected graph stored as an adjacency list.

D. Once you have constructed the new graph, how do you find the shortest fair path for Alice, Bob, and Carol? What algorithm do you use? How do you apply that algorithm to the new graph? How do you compute the solution?

E. What is the running time of your complete algorithm? Assume that the map contains n nodes and m road segments (each one hour long), and that it is a weighted, connected, directed graph. Explain in one sentence.

F. On the next trip, Alice, Bob, and Carol are joined by all their friends. Now there are k friends going on a road trip, trying to divide up the driving exactly equally. Explain how your algorithm generalizes from 3 friends to k friends. How would you find the shortest path that ensures each of the k friends drives an equal number of hours? What is the running time of your new algorithm as a function of n, m, and k?

G. Alice notices that, strangely, there are no cycles on the map. (All the roads are one way?) How would you improve the algorithm? What is the running time of the improved algorithm? Again, assume there are k friends and give your answer as a function of n, m, and k.

Question 11: Wrap Up [3 marks]

A. What was the most interesting thing you learned in this class? (Be specific.) [1 mark]

B. What was the most important thing you learned in this class? (Be specific.) [1 mark]

C. What do you think you will remember from this class in five years? (Be specific.) [1 mark]

Question 12: Just for fun: The Blue-Eyed Islanders [0 marks]

[0 mark]

There is an island upon which a tribe resides. The tribe consists of 1000 people, with various eye colours. Yet, their religion forbids them from knowing their own eye color, or even from discussing the topic. Thus, each resident can (and does) see the eye colors of all other residents, but has no way of discovering his or her own (there are no reflective surfaces).

If a tribesperson does discover his or her own eye color, then their religion compels them to say goodbye to every other islander and leave the island forever by noon the following day. All the tribespeople are highly logical and devout, and they all know that all the others are also highly logical and devout (and they all know that they all know that all the others are highly logical and devout, and so forth).

Of the 1000 islanders, it turns out that 100 of them have blue eyes and 900 of them have brown eyes, although the islanders are not initially aware of these statistics (each of them can of course only see 999 of the 1000 tribespeople).

One day, a blue-eyed foreigner visits the island and wins the complete trust of the tribe. One evening, he addresses the entire tribe to thank them for their hospitality. However, not knowing the customs, the foreigner makes the mistake of mentioning eye color in his address, remarking "how unusual it is to see another blue-eyed person like myself in this region of the world".

What effect, if anything, does this faux pas have on the tribe?

Courtesy of Terence Tao:
http://terrytao.wordpress.com/2011/04/07/the-blue-eyed-islanders-puzzle-repost/.

Nothing to submit for this question. This is for you to think about after the exam. You can post your thoughts and ideas in the class forum.

— E N D O F P A P E R —

Scratch Paper

— H A P P Y H O L I D A Y S ! —

CS2040S — Data Structures and Algorithms
School of Computing
National University of Singapore

Final Assessment — Answer Sheet

2021/2022 Semester 2

Time allowed: 2 hours

Instructions (please read carefully):

A. Write your **student number** on the right and, using ink or pencil, shade the corresponding circle **completely** in the grid for each digit or letter. Do not write your name.

B. This answer booklet comprises **SIXTEEN (16) pages**, including this cover page.

C. All questions must be answered in the space provided; no extra sheets will be accepted as answers. You may use the extra page behind this cover page if you need more space for your answers.

D. You must submit only the **ANSWER SHEET** and no other documents. The question set may be used as scratch paper.

E. An excerpt of the question may be provided to aid you in answering in the correct box. It is not the exact question. You should still refer to the original question in the question booklet.

F. You are allowed to use pencils, ball-pens or fountain pens, as you like as long as it is legible (no red color, please).

G. **Marks may be deducted** for i) unrecognisable handwriting, and/or ii) excessively long explanations.

H. Each multiple choice question is intended to have only one answer. Please fill in the appropriate bubble.

STUDENT NUMBER

A									
U ○	0	0	0	0	0	0	0	A	N
A ●	1	1	1	1	1	1	1	B	R
HT ○	2	2	2	2	2	2	2	E	U
NT ○	3	3	3	3	3	3	3	H	W
	4	4	4	4	4	4	4	J	X
	5	5	5	5	5	5	5	L	Y
	6	6	6	6	6	6	6	M	
	7	7	7	7	7	7	7		
	8	8	8	8	8	8	8		
	9	9	9	9	9	9	9		

For Examiner's Use Only

Question	Marks
Q1	/ 7
Q2	/ 6
Q3	/ 5
Q4	/ 11
Q5	/ 9
Q6	/ 6
Q7	/ 10
Q8	/ 18
Q9	/ 10
Q10	/ 15
Q11	/ 3
Q12	/ 0
Total	/100

This page is intentionally left blank.
Use it ONLY if you need extra space for your answers, and indicate the **question number clearly** as well as in the original answer box. **Do NOT** use it for your rough work.

Question 1A Binary search is correct with repeated values. [1 marks]

◯ True	◯ False

Question 1B Peak finding to find maximum of $f+g$. [1 marks]

◯ True	◯ False

Question 1C Recurrence is correct? [1 marks]

◯ True	◯ False

Question 1D InsertionSort on modified array in $O(n)$ time? [1 marks]

◯ True	◯ False

Question 1E Amortized symbol table. [1 marks]

◯ True (Frank is correct)	◯ False (Frank is wrong)

Question 1F Bellman-Ford property. [1 marks]

◯ True	◯ False

Question 1G Dijkstra's modification works correctly? [1 marks]

◯ True	◯ False

Question 2A Use BFS instead of DFS to find a shortest path. [1 marks]

◯ Performance	◯ Correctness	◯ Equally good	◯ Wrong choice

Question 2B Use adjacency matrix representation for sparse graphs. [1 marks]

◯ Performance	◯ Correctness	◯ Equally good	◯ Wrong choice

Question 2C Sort edges by weight in Kruskal's. [1 marks]

◯ Performance	◯ Correctness	◯ Equally good	◯ Wrong choice

Question 2D Use BFS instead of Bellman-Ford in a graph with identical edge weights. [1 marks]

◯ Performance	◯ Correctness	◯ Equally good	◯ Wrong choice

Question 2E Post-order DFS to find a topological ordering. [1 marks]

○ Performance ○ Correctness ○ Equally good ○ Wrong choice

Question 2F Post-order DFS to find a topological ordering. [1 marks]

○ Performance ○ Correctness ○ Equally good ○ Wrong choice

Question 3A Estimate is equal to distance on some path. [1 marks]

○ Bellman-Ford
○ Dijkstra
○ Both Bellman-Ford and Dijkstra
○ Neither Bellman-Ford nor Dijkstra

Question 3B Each edge is relaxed at most once. [1 marks]

○ Bellman-Ford
○ Dijkstra
○ Both Bellman-Ford and Dijkstra
○ Neither Bellman-Ford nor Dijkstra

Question 3C Estimate is $\geq$ distance. [1 marks]

○ Bellman-Ford
○ Dijkstra
○ Both Bellman-Ford and Dijkstra
○ Neither Bellman-Ford nor Dijkstra

Question 3D Estimate is $\leq$ distance. [1 marks]

○ Bellman-Ford
○ Dijkstra
○ Both Bellman-Ford and Dijkstra
○ Neither Bellman-Ford nor Dijkstra

Question 3E Safe to terminate early? [1 marks]

- ◯ Bellman-Ford
- ◯ Dijkstra
- ◯ Both Bellman-Ford and Dijkstra
- ◯ Neither Bellman-Ford nor Dijkstra

Question 4A Terminates? [2 marks]

◯ True (always terminates) ◯ False (does not always terminate)

Question 4B At most once? [1 marks]

◯ True ◯ False

Question 4C More than once? [1 marks]

◯ True ◯ False

Question 4D At most k times in the queue? [1 marks]

◯ True ◯ False

Question 4E What does the algorithm guarantee? [3 marks]

- ◯ The length of the (unweighted) shortest path from the source to v.
- ◯ The length of the (unweighted) longest path from the source to v.
- ◯ The number of distinct paths from the source to v.
- ◯ None of the above.

Question 4F Asymptotic running time? [3 marks]

- ◯ $O(n)$
- ◯ $O(n+m)$
- ◯ $O(nm$
- ◯ $O(nm^2)$
- ◯ $O(2^{2m})$
- ◯ None of the above.

Question 5A Fill in the blank. [1 marks]

- ◯ `POSITIVE_INFINITY`
- ◯ `NEGATIVE_INFINITY`
- ◯ `0`
- ◯ `1`
- ◯ `-1`

Question 5B Fill in the blank. [1 marks]

- ◯ `POSITIVE_INFINITY`
- ◯ `NEGATIVE_INFINITY`
- ◯ `0`
- ◯ `1`
- ◯ `-1`

Question 5C Fill in the blank. [1 marks]

- ◯ `PriorityQueue ds = new PriorityQueue();`
- ◯ `Queue ds = new Queue();`
- ◯ `Stack ds = new Stack();`
- ◯ `int ds[] = new int[n];`

Question 5D Fill in the blank. [1 marks]

- ◯ `ds.insert(v, est[v])`
- ◯ `ds.increaseKey(v, est[v])`
- ◯ `ds.decreaseKey(v, est[v])`
- ◯ `ds.extractMin()`
- ◯ `ds.extractMax()`
- ◯ `ds.enqueue(v)`
- ◯ `ds.dequeue()`
- ◯ `ds.push(v)`
- ◯ `ds.pop()`
- ◯ `ds[0]`
- ◯ `ds[v] = est[v]`
- ◯ `ds[s]`

Question 5E Fill in the blank. [1 marks]

- ◯ `ds.insert(v, est[v])`
- ◯ `ds.increaseKey(v, est[v])`
- ◯ `ds.decreaseKey(v, est[v])`
- ◯ `ds.extractMin()`
- ◯ `ds.extractMax()`
- ◯ `ds.enqueue(v)`
- ◯ `ds.dequeue()`
- ◯ `ds.push(v)`
- ◯ `ds.pop()`
- ◯ `ds[0]`
- ◯ `ds[v] = est[v]`
- ◯ `ds[s]`

Question 5F Fill in the blank. [1 marks]

- ◯ `est[u] < est[v] + G.weight(u,v)`
- ◯ `est[v] < est[u] + G.weight(u,v)`
- ◯ `est[v] > est[u] + G.weight(u,v)`
- ◯ `est[u] > est[v] + G.weight(u,v)`
- ◯ `est[u] == est[v] + G.weight(u,v)`
- ◯ `est[u] + G.weight(u,v)`
- ◯ `est[v] + G.weight(u,v)`
- ◯ `G.weight(u,v)`

Question 5G Fill in the blank. [1 marks]

- ◯ `est[u] < est[v] + G.weight(u,v)`
- ◯ `est[v] < est[u] + G.weight(u,v)`
- ◯ `est[v] > est[u] + G.weight(u,v)`
- ◯ `est[u] > est[v] + G.weight(u,v)`
- ◯ `est[u] == est[v] + G.weight(u,v)`
- ◯ `est[u] + G.weight(u,v)`
- ◯ `est[v] + G.weight(u,v)`
- ◯ `G.weight(u,v)`

Question 5H Fill in the blank. [1 marks]

- ◯ `ds.insert(v, est[v])`
- ◯ `ds.increaseKey(v, est[v])`
- ◯ `ds.decreaseKey(v, est[v])`
- ◯ `ds.extractMin()`
- ◯ `ds.extractMax()`
- ◯ `ds.enqueue(v)`
- ◯ `ds.dequeue()`
- ◯ `ds.push(v)`
- ◯ `ds.pop()`
- ◯ `ds[0]`
- ◯ `ds[v] = est[v]`
- ◯ `ds[s]`

Question 5I How to implement parent in shortest path tree? [1 marks]

- ◯ Add the line `parent[u] = v` immediately after line 24.
- ◯ Add the line `parent[u] = v` immediately after line 29.
- ◯ Add the line `parent[v] = u` immediately after line 24.
- ◯ Add the line `parent[v] = u` immediately after line 29.
- ◯ None of the above options will yield a correct shortest path tree.

Question 6A Asymptotic running time of BFS on G? [2 marks]

- ◯ $O(n)$
- ◯ $O(n+m)$
- ◯ $O(m \log n)$
- ◯ $O(n^2)$.
- ◯ $O(mn)$.
- ◯ $O(n^3)$.
- ◯ $O(mn^2)$.

Question 6B Asymptotic running time of Prim's Algorithm on G? [2 marks]

- ◯ $O(n)$
- ◯ $O(n+m)$
- ◯ $O(m \log n)$
- ◯ $O(n^2)$
- ◯ $O(mn)$
- ◯ $O(n^3)$
- ◯ $O(mn^2)$

Question 6C Asymptotic running time of Dijkstra's Algorithm using the SuperQueue? [2 marks]

- ◯ $O(m+n)$
- ◯ $O(n \log^2 n + m \log n)$
- ◯ $O(m \log n)$
- ◯ $O(n^2)$
- ◯ $O(n^2 + m \log n)$
- ◯ $O(n^2 \log n + m \log n)$
- ◯ $O(m \log^2 n + n \log n)$
- ◯ $O(mn)$
- ◯ $O(mn^2)$

Question 7A Subway design? [2 marks]

- ◯ Breadth-First Search (BFS)
- ◯ Bellman-Ford
- ◯ Dijkstra's Algorithm
- ◯ Prim's Algorithm
- ◯ DAG-shortest-paths
- ◯ Floyd-Warshall
- ◯ Travelling Salesman Approximation (from Problem Set 8)
- ◯ Cannot be solved efficiently.

Question 7B Spinout? [2 marks]

- ◯ Breadth-First Search (BFS)
- ◯ Bellman-Ford
- ◯ Dijkstra's Algorithm
- ◯ Prim's Algorithm
- ◯ DAG-shortest-paths
- ◯ Floyd-Warshall
- ◯ Travelling Salesman Approximation (from Problem Set 8)

Question 7C Taxi route? [2 marks]

- ◯ Breadth-First Search (BFS)
- ◯ Bellman-Ford
- ◯ Dijkstra's Algorithm
- ◯ Prim's Algorithm
- ◯ DAG-shortest-paths
- ◯ Floyd-Warshall
- ◯ Travelling Salesman Approximation (from Problem Set 8)
- ◯ Cannot be solved efficiently.

Question 7D Shipping route? [2 marks]

- ◯ Breadth-First Search (BFS)
- ◯ Bellman-Ford
- ◯ Dijkstra's Algorithm
- ◯ Prim's Algorithm
- ◯ DAG-shortest-paths
- ◯ Floyd-Warshall
- ◯ Travelling Salesman Approximation (from Problem Set 8)
- ◯ Cannot be solved efficiently.

Question 7E Shipping route again? [2 marks]

- ◯ Breadth-First Search (BFS)
- ◯ Bellman-Ford
- ◯ Dijkstra's Algorithm
- ◯ Prim's Algorithm
- ◯ DAG-shortest-paths
- ◯ Floyd-Warshall
- ◯ Travelling Salesman Approximation (from Problem Set 8)
- ◯ Cannot be solved efficiently.

Question 8A Strongly connected components? [2 marks]

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6
- ◯ 7
- ◯ 8 or more

Question 8B Topological order? [2 marks]

- ◯ It has a unique topological order.
- ◯ It has a topological order, but it is not unique.
- ◯ It does not have a topological order, but if we delete one edge then it will have a topological order.
- ◯ It does not have a topological order, and we need to delete more than one edge for it to have a topological order.

Question 8C DFS, seventh node visited? [2 marks]

- ◯ A
- ◯ B
- ◯ C
- ◯ D
- ◯ E
- ◯ F
- ◯ G
- ◯ H
- ◯ I
- ◯ J
- ◯ K
- ◯ L

Question 8D BFS, seventh node visited? [2 marks]

- ◯ A
- ◯ B
- ◯ C
- ◯ D
- ◯ E
- ◯ F
- ◯ G
- ◯ H
- ◯ I
- ◯ J
- ◯ K
- ◯ L

Question 8E Dijkstra, fourth node? [2 marks]

◯ A	◯ E	◯ I
◯ B	◯ F	◯ J
◯ C	◯ G	◯ K
◯ D	◯ H	◯ L

Question 8F Kruskal's, first edge not included? [2 marks]

◯ 1	◯ 7	◯ 12
◯ 2	◯ 8	
◯ 3	◯ 8	◯ 13
◯ 4	◯ 9	
◯ 5	◯ 10	◯ 14
◯ 6	◯ 11	◯ 15 or above.

Question 8G Prim's, sixth node? [2 marks]

◯ A	◯ E	◯ I
◯ B	◯ F	◯ J
◯ C	◯ G	◯ K
◯ D	◯ H	◯ L

Question 8H edge (J,H) is in the MST. [1 marks]

◯ True ◯ False

Question 8I edge (B,C) is <u>not</u> in the MST. [1 marks]

◯ True ◯ False

Question 8J edge (J,L) is <u>not</u> in the MST. [1 marks]

◯ True ◯ False

Question 8K edge (H,E) is <u>not</u> in the MST. [1 marks]

◯ True ◯ False

Question 9A Number of elements in longest linked list? [3 marks]

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6
- ◯ None of the above.

Question 9B Last cell explored in search for ‘F’? [3 marks]

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6
- ◯ 7
- ◯ 8
- ◯ 9
- ◯ 10
- ◯ 11
- ◯ 12
- ◯ 13

Question 9C Every insert succeeds? [1 marks]

- ◯ Chaining
- ◯ Open addressing
- ◯ Both
- ◯ Neither

Question 9D Worst-case searches? [1 marks]

- ◯ Chaining
- ◯ Open addressing
- ◯ Both
- ◯ Neither

Question 9E Expected cost searches? [1 marks]

- ◯ Chaining
- ◯ Open addressing
- ◯ Both
- ◯ Neither

Question 9F Chaining asymptotically faster than open addressing, in expectation.[1 marks]

◯ True ◯ False

Question 10A

[?? marks]

What are the nodes in the new graph?:

What are the edges in the new graph?:

Question 10B

[?? marks]

Transformed graph:

Question 10C

[?? marks]

Asymptotic running time as a function of n and m:

Question 10D

[?? marks]

Which algorithm? (< 5 words)

How to use the algorithm to find the solution?

Question 10E

[?? marks]

Running time:

Explanation:

Question 10F

[?? marks]

Generalized version, running time:

Generalized version, explanation (one sentence only):

Question 10G

[?? marks]

No cycles, Running time:

No cycles, explanation (one sentence only):

Question 11A

[1 marks]

What was the most interesting thing you learned in this class? (Be specific.)

Question 11B

[1 marks]

What was the most important thing you learned in this class? (Be specific.)

Question 11C

[1 marks]

What do you think you will remember from this class in five years? (Be specific.)

— END OF ANSWER SHEET —

Question 1A Binary search is correct with repeated values. [1 marks]

◯ True ◯ False True

Question 1B Peak finding to find maximum of $f+g$. [1 marks]

◯ True ◯ False False (there is more than one peak, so peak finding will not find a max)

Question 1C Recurrence is correct? [1 marks]

◯ True ◯ False False

Question 1D InsertionSort on modified array in $O(n)$ time? [1 marks]

◯ True ◯ False True

Question 1E Amortized symbol table. [1 marks]

◯ True (Frank is correct) ◯ False (Frank is wrong) True (this is essentially the definition of amortized complexity)

Question 1F Bellman-Ford property. [1 marks]

◯ True ◯ False False

Question 1G Dijkstra's modification works correctly? [1 marks]

◯ True ◯ False False

Question 2A Use BFS instead of DFS to find a shortest path. [1 marks]

◯ Performance ◯ Correctness ◯ Equally good ◯ Wrong choice Correctness (DFS does not find shortest paths)

Question 2B Use adjacency matrix representation for sparse graphs. [1 marks]

◯ Performance ◯ Correctness ◯ Equally good ◯ Wrong choice Wrong Choice (use an adjacency list if graph is sparse)

Question 2C Sort edges by weight in Kruskal's. [1 marks]

◯ Performance ◯ Correctness ◯ Equally good ◯ Wrong choice Correctness (arbitrary order results in non-minimum spanning tree)

Question 2D Use BFS instead of Bellman-Ford in a graph with identical edge weights. [1 marks]

◯ Performance ◯ Correctness ◯ Equally good ◯ Wrong choice Performance (BFS is faster)

Question 2E Post-order DFS to find a topological ordering. [1 marks]

◯ Performance ◯ Correctness ◯ Equally good ◯ Wrong choice Equally good (both are equally correct and equally fast)

Question 2F Post-order DFS to find a topological ordering. [1 marks]

◯ Performance ◯ Correctness ◯ Equally good ◯ Wrong choice Correctness (a queue will not result in a correct implementation of Prim's)

Question 3A Estimate is equal to distance on some path. [1 marks]

◯ Bellman-Ford
◯ Dijkstra
◯ Both Bellman-Ford and Dijkstra
◯ Neither Bellman-Ford nor Dijkstra

Both (prove by induction)

Question 3B Each edge is relaxed at most once. [1 marks]

◯ Bellman-Ford
◯ Dijkstra
◯ Both Bellman-Ford and Dijkstra
◯ Neither Bellman-Ford nor Dijkstra

Dijkstra (Bellman-Ford relaxes edges more than once)

Question 3C Estimate is $\geq$ distance. [1 marks]

◯ Bellman-Ford
◯ Dijkstra
◯ Both Bellman-Ford and Dijkstra
◯ Neither Bellman-Ford nor Dijkstra

Both (this is a key invariant for both)

Question 3D Estimate is $\leq$ distance. [1 marks]

- ◯ Bellman-Ford
- ◯ Dijkstra
- ◯ Both Bellman-Ford and Dijkstra
- ◯ Neither Bellman-Ford nor Dijkstra

Neither

Question 3E Safe to terminate early? [1 marks]

- ◯ Bellman-Ford
- ◯ Dijkstra
- ◯ Both Bellman-Ford and Dijkstra
- ◯ Neither Bellman-Ford nor Dijkstra

Neither (Dijkstra can terminate early when destination is extracted from PQ.)

Question 4A Terminates? [2 marks]

◯ True (always terminates) ◯ False (does not always terminate) True (only because it is a DAG; note it does not check visited)

Question 4B At most once? [1 marks]

◯ True ◯ False False (does not check visited, so nodes are re-added)

Question 4C More than once? [1 marks]

◯ True ◯ False True

Question 4D At most k times in the queue? [1 marks]

◯ True ◯ False False

Question 4E What does the algorithm guarantee? [3 marks]

- ◯ The length of the (unweighted) shortest path from the source to v.
- ◯ The length of the (unweighted) longest path from the source to v.
- ◯ The number of distinct paths from the source to v.
- ◯ None of the above.

Longest path OR none of the above (because est not initialized)

Question 4F Asymptotic running time? [3 marks]

- ○ $O(n)$
- ○ $O(n+m)$
- ○ $O(nm$
- ○ $O(nm^2)$
- ○ $O(2^{2m})$
- ○ None of the above.

$O(2^{2m})$ (because exponential time to check all paths in the graph, which it does)

Question 5A Fill in the blank. [1 marks]

- ○ `POSITIVE_INFINITY`
- ○ `NEGATIVE_INFINITY`
- ○ `0`
- ○ `1`
- ○ `-1`

Positive Infinity

Question 5B Fill in the blank. [1 marks]

- ○ `POSITIVE_INFINITY`
- ○ `NEGATIVE_INFINITY`
- ○ `0`
- ○ `1`
- ○ `-1`

Zero

Question 5C Fill in the blank. [1 marks]

- ○ `PriorityQueue ds = new PriorityQueue();`
- ○ `Queue ds = new Queue();`
- ○ `Stack ds = new Stack();`
- ○ `int ds[] = new int[n];`

Priority Queue

Question 5D Fill in the blank. [1 marks]

- ◯ `ds.insert(v, est[v])`
- ◯ `ds.increaseKey(v, est[v])`
- ◯ `ds.decreaseKey(v, est[v])`
- ◯ `ds.extractMin()`
- ◯ `ds.extractMax()`
- ◯ `ds.enqueue(v)`
- ◯ `ds.dequeue()`
- ◯ `ds.push(v)`
- ◯ `ds.pop()`
- ◯ `ds[0]`
- ◯ `ds[v] = est[v]`
- ◯ `ds[s]`

ds.insert(v, est[v])

Question 5E Fill in the blank. [1 marks]

- ◯ `ds.insert(v, est[v])`
- ◯ `ds.increaseKey(v, est[v])`
- ◯ `ds.decreaseKey(v, est[v])`
- ◯ `ds.extractMin()`
- ◯ `ds.extractMax()`
- ◯ `ds.enqueue(v)`
- ◯ `ds.dequeue()`
- ◯ `ds.push(v)`
- ◯ `ds.pop()`
- ◯ `ds[0]`
- ◯ `ds[v] = est[v]`
- ◯ `ds[s]`

ds.extractMin()

Question 5F Fill in the blank. [1 marks]

- ◯ `est[u] < est[v] + G.weight(u,v)`
- ◯ `est[v] < est[u] + G.weight(u,v)`
- ◯ `est[v] > est[u] + G.weight(u,v)`
- ◯ `est[u] > est[v] + G.weight(u,v)`
- ◯ `est[u] == est[v] + G.weight(u,v)`
- ◯ `est[u] + G.weight(u,v)`
- ◯ `est[v] + G.weight(u,v)`
- ◯ `G.weight(u,v)`

est[v] > est[u] + G.weight(u,v)

Question 5G Fill in the blank. [1 marks]

- ◯ `est[u] < est[v] + G.weight(u,v)`
- ◯ `est[u] == est[v] + G.weight(u,v)`
- ◯ `est[v] < est[u] + G.weight(u,v)`
- ◯ `est[u] + G.weight(u,v)`
- ◯ `est[v] > est[u] + G.weight(u,v)`
- ◯ `est[v] + G.weight(u,v)`
- ◯ `est[u] > est[v] + G.weight(u,v)`
- ◯ `G.weight(u,v)`

est[u] + G.weight(u,v)

Question 5H Fill in the blank. [1 marks]

- ◯ `ds.insert(v, est[v])`
- ◯ `ds.dequeue()`
- ◯ `ds.increaseKey(v, est[v])`
- ◯ `ds.push(v)`
- ◯ `ds.decreaseKey(v, est[v])`
- ◯ `ds.pop()`
- ◯ `ds.extractMin()`
- ◯ `ds[0]`
- ◯ `ds.extractMax()`
- ◯ `ds[v] = est[v]`
- ◯ `ds.enqueue(v)`
- ◯ `ds[s]`

ds.decreaseKey(v, est[v])

Question 5I How to implement parent in shortest path tree? [1 marks]

- ◯ Add the line `parent[u] = v` immediately after line 24.
- ◯ Add the line `parent[u] = v` immediately after line 29.
- ◯ Add the line `parent[v] = u` immediately after line 24.
- ◯ Add the line `parent[v] = u` immediately after line 29.
- ◯ None of the above options will yield a correct shortest path tree.

Add the line parent[v] = u immediately after line 24.

Question 6A Asymptotic running time of BFS on G? [2 marks]

- ◯ $O(n)$
- ◯ $O(mn)$.
- ◯ $O(n+m)$
- ◯ $O(m \log n)$
- ◯ $O(n^3)$.
- ◯ $O(n^2)$.
- ◯ $O(mn^2)$.

$O(n^2)$

Question 6B Asymptotic running time of Prim's Algorithm on G? [2 marks]

- ○ $O(n)$
- ○ $O(n+m)$
- ○ $O(m\log n)$
- ○ $O(n^2)$
- ○ $O(mn)$
- ○ $O(n^3)$
- ○ $O(mn^2)$

$O(mn)$ (in fact, $O(n^2+m\log n)$)

Question 6C Asymptotic running time of Dijkstra's Algorithm using the SuperQueue? [2 marks]

- ○ $O(m+n)$
- ○ $O(n\log^2 n+m\log n)$
- ○ $O(m\log n)$
- ○ $O(n^2)$
- ○ $O(n^2+m\log n)$
- ○ $O(n^2\log n+m\log n)$
- ○ $O(m\log^2 n+n\log n)$
- ○ $O(mn)$
- ○ $O(mn^2)$

$O(n^2+m\log n)$ (i.e., n*insert + n*extractMin + m*decreaseKey)

Question 7A Subway design? [2 marks]

- ○ Breadth-First Search (BFS)
- ○ Bellman-Ford
- ○ Dijkstra's Algorithm
- ○ Prim's Algorithm
- ○ DAG-shortest-paths
- ○ Floyd-Warshall
- ○ Travelling Salesman Approximation (from Problem Set 8)
- ○ Cannot be solved efficiently.

Prim's Algorithm (since the goal is to minimize tunnel length, not shortest paths)

Question 7B Spinout? [2 marks]

- ○ Breadth-First Search (BFS)
- ○ Bellman-Ford
- ○ Dijkstra's Algorithm
- ○ Prim's Algorithm
- ○ DAG-shortest-paths
- ○ Floyd-Warshall
- ○ Travelling Salesman Approximation (from Problem Set 8)

BFS (since we are searching for the shortest path in the state-space of a puzzle)

Question 7C Taxi route? [2 marks]

- ◯ Breadth-First Search (BFS)
- ◯ Bellman-Ford
- ◯ Dijkstra's Algorithm
- ◯ Prim's Algorithm
- ◯ DAG-shortest-paths
- ◯ Floyd-Warshall
- ◯ Travelling Salesman Approximation (from Problem Set 8)
- ◯ Cannot be solved efficiently.

Dijkstra, repeated for every every source. (The question asks for the diameter of the road network graph. If the road network is dense, then Floyd-Warshall is best, but the network is sparse so Dijkstra is better.)

Question 7D Shipping route? [2 marks]

- ◯ Breadth-First Search (BFS)
- ◯ Bellman-Ford
- ◯ Dijkstra's Algorithm
- ◯ Prim's Algorithm
- ◯ DAG-shortest-paths
- ◯ Floyd-Warshall
- ◯ Travelling Salesman Approximation (from Problem Set 8)
- ◯ Cannot be solved efficiently.

Cannot be solved efficiently (Longest path cannot be solved efficiently).

Question 7E Shipping route again? [2 marks]

- ◯ Breadth-First Search (BFS)
- ◯ Bellman-Ford
- ◯ Dijkstra's Algorithm
- ◯ Prim's Algorithm
- ◯ DAG-shortest-paths
- ◯ Floyd-Warshall
- ◯ Travelling Salesman Approximation (from Problem Set 8)
- ◯ Cannot be solved efficiently.

Cannot be solved efficiently (Longest path cannot be solved efficiently).

Question 8A Strongly connected components? [2 marks]

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6
- ◯ 7
- ◯ 8 or more

4

Question 8B Topological order? [2 marks]

- ◯ It has a unique topological order.
- ◯ It has a topological order, but it is not unique.
- ◯ It does not have a topological order, but if we delete one edge then it will have a topological order.
- ◯ It does not have a topological order, and we need to delete more than one edge for it to have a topological order.

It does not have a topological order (because of cycles) and we need to delete more than one edge to remove all cycles.

Question 8C DFS, seventh node visited? [2 marks]

◯ A	◯ E	◯ I
◯ B	◯ F	◯ J
◯ C	◯ G	◯ K
◯ D	◯ H	◯ L

D (DFS sequence: A, K, L, B, C, E, D; if adjacency lists are sorted by weight: A, L, B, C, G, H, F

Question 8D BFS, seventh node visited? [2 marks]

◯ A	◯ E	◯ I
◯ B	◯ F	◯ J
◯ C	◯ G	◯ K
◯ D	◯ H	◯ L

G (BFS sequence: A, K, L, B, C, E, G; if the edge lists are sorted by weight, then the BFS sequence is: A, L, K, B, C, G, E)

Question 8E Dijkstra, fourth node? [2 marks]

◯ A	◯ E	◯ I
◯ B	◯ F	◯ J
◯ C	◯ G	◯ K
◯ D	◯ H	◯ L

K (sequence: A, L, B, K)

Question 8F Kruskal's, first edge not included? [2 marks]

◯ 1 ◯ 7 ◯ 12
◯ 2 ◯ 8
◯ 3 ◯ 8 ◯ 13
◯ 4 ◯ 9
◯ 5 ◯ 10 ◯ 14
◯ 6 ◯ 11 ◯ 15 or above.

9 (edges 1 through 8 are added, while 9 is not included because of the cycle (B, A, L))

Question 8G Prim's, sixth node? [2 marks]

◯ A ◯ E ◯ I
◯ B ◯ F ◯ J
◯ C ◯ G ◯ K
◯ D ◯ H ◯ L

H (order: A, L, B, K, J, H)

Question 8H edge (J,H) is in the MST. [1 marks]

◯ True ◯ False True (e.g., the left half and right half of the graph)

Question 8I edge (B,C) is not in the MST. [1 marks]

◯ True◯ False False (it is the heaviest edge across the cut, but that is not a good reason to exclude an edge from an MST)

Question 8J edge (J,L) is not in the MST. [1 marks]

◯ True◯ False False (it is not in the MST, but that is not a good reason to exclude an edge from an MST since it is possible that every outgoing edge can be in an MST, e.g., a star

Question 8K edge (H,E) is not in the MST. [1 marks]

◯ True ◯ False True (this is not in the MST, and this is a godo reason why it is not in the MST)

Question 9A Number of elements in longest linked list? [3 marks]

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6
- ◯ None of the above.

3 (E, C, and K are all mapped to slot 5.)

Question 9B Last cell explored in search for 'F'? [3 marks]

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5
- ◯ 6
- ◯ 7
- ◯ 8
- ◯ 9
- ◯ 10
- ◯ 11
- ◯ 12
- ◯ 13

13 (The table looks like (G, empty, empty, O, E, T, S, C, K, R, A, P, empty). F maps to 9, and cells 9, 10, 11, and 12 are full, so 13 is the answer.)

Question 9C Every insert succeeds? [1 marks]

- ◯ Chaining
- ◯ Open addressing
- ◯ Both
- ◯ Neither

Chaining. With open addressing, the table may be full and then insertions will fail.

Question 9D Worst-case searches? [1 marks]

- ◯ Chaining
- ◯ Open addressing
- ◯ Both
- ◯ Neither

Neither. (Both, in the worst-case, can result in $\Theta(n)$ search operations.)

Question 9E Expected cost searches? [1 marks]

◯ Chaining
◯ Open addressing
◯ Both
◯ Neither

Both. (Under good hashing assumptions, both guarantee expect $O(1)$ operations.)

Question 9F Chaining asymptotically faster than open addressing, in expectation.[1 marks]

◯ True◯ False True. (For hashing with chaining, operations have cost a tad less than 1, for this load. For open adressing, operations have a cost a little more than $\log n$, with this load.)

Question 10A [?? marks]

What are the nodes in the new graph?:

Replicate each node in the graph 3 times so that the state of each node represents the pair (B,x) where B is the original city and x is the number of hours driven modulo 3.

What are the edges in the new graph?:

Each edge (B,C) is replaced with edges $((B,x),(B,x+1 \texttt{mod} 3))$.

Question 10B

[?? marks]

Transformed graph:

Pretty picture goes here.

Question 10C

[?? marks]

Asymptotic running time as a function of n and m:

$O(n+m)$ or $O(m)$.

Question 10D

[?? marks]

Which algorithm? (< 5 words)

Use Dijkstra's algorithm.

How to use the algorithm to find the solution?

Run Dijkstra's Algorithm starting at A(lberta). Use it to find the shortest path to node $(Z,0)$, i.e., node Z where the number of hours modulo 3 is zero.

Question 10E

[?? marks]

Running time:

$O(m \log n)$.

Explanation:

The constructed graph has $O(n)$ nodes and $O(m)$ edges and we only need to run Dijkstra's once to find the solution.

Question 10F

[?? marks]

Generalized version, running time:

$O(km \log nk)$.

Generalized version, explanation (one sentence only):

The same solution works, except now each node and each is replicated k times.

Question 10G

[?? marks]

No cycles, Running time:

$O(km)$.

No cycles, explanation (one sentence only):

Since the graph is a DAG, we can now use the DAG_SSSP algorithm (i.e., toposort and relax the edges in order).

Question 11A
[1 marks]

What was the most interesting thing you learned in this class? (Be specific.)

Question 11B
[1 marks]

What was the most important thing you learned in this class? (Be specific.)

Question 11C
[1 marks]

What do you think you will remember from this class in five years? (Be specific.)