Data Structures and Algorithms 31251
Assessment Task 3 – Alternative Examination
Autumn 2022

Instructions

This examination is made available online at **9:30am** on **July 13 2022**.

Your completed answer file is due at **9:30am** on **July 14 2022** and must be submitted online via https://canvas.uts.edu.au/courses/22030/assignments/117367 on Canvas.

There are **15 multiple-choice questions and 6 short-answer questions**. Your answer to each question attempted should commence after the question number in the answer file.

The examination is worth **30%** of the marks available in this subject. The contribution each question makes to the total examination mark is indicated in marks.

This examination is an open book examination.

This examination is expected to take approximately **2 hours** of working time. You are advised to allocate your time accordingly. Your answer file may be submitted at any time before the due time. Please allow time to complete the submission process.

Please submit your file in PDF/Word format unless directed otherwise. Please name your file as follows:

EXAM_31251_student number e.g. EXAM_31251_12345678

Word Limit

There is a word limit for each question. The most important thing is to answer the question in a succinct manner. This means that your answer can consist of a word count less than the imposed word limit. A ten percent (10%) leeway on word counts is permitted.

Your answers should not contain any footnote/reference.

Important Notice – Exam Conditions and Academic Integrity

In attempting this examination and submitting an answer file, candidates are undertaking that the work they submit is a result of their own unaided efforts and that they have not discussed the questions or possible answers with other persons during the examination period. Candidates who are found to have participated in any form of cooperation or collusion or any activity which could amount to academic misconduct in the

answering of this examination will have their marks withdrawn and disciplinary action will be initiated on a complaint from the Examiner.

Exam answers must be submitted **via Turnitin**. Staff may ask that a student undertake an oral test to ensure they have completed the work on their own and to assess their knowledge of the answers they have submitted.

Students must not post any requests for clarification on the Discussion Boards on Blackboard, Canvas or Microsoft Teams. Any requests for clarification should be directed by email to **Xianzhi Wang** on **Xianzhi.Wang@uts.edu.au**. Where clarification is required it will be broadcast by email to all students in the exam group. **Xianzhi Wang** will be available via email during the first hour and the last hour of the examination time window.

Question 1 (0.8 Mark)

Suppose `x` is the value at the front of a `std::queue` after the following operations: `push(5)`, `push(10)`, `push(9)`, `pop()`, `push(1)`, `push(1)`, `pop()`, `push(6)`, `pop()`, `pop()`. What operation sequences make the value at the top of a `std::stack` equal `x` (More than one answer may be selected)?

A. `push(5)`, `push(10)`, `pop()`, `push(9)`, `push(1)`, `pop()`, `push(1)`, `push(6)`, `pop()`
B. `push(5)`, `push(10)`, `push(9)`, `push(1)`, `pop()`, `pop()`, `pop()`, `push(1)`, `push(6)`
C. `push(5)`, `push(10)`, `push(9)`, `pop()`, `pop()`, `push(1)`, `push(6)`, `push(1)`, `pop()`
D. `push(5)`, `pop()`, `push(10)`, `push(1)`, `push(9)`, `pop()`, `push(1)`, `push(6)`, `pop()`

Question 2 (0.8 Mark)

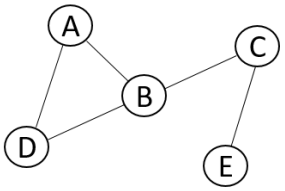
What is the time complexity of the following code (Suppose `m>0` and `n>1`)?

```
int func (int m, int n){
    int i=0;
    while (i<m){
        i++;
    }
    for (; i<n-1; i++){
        for (int j=i+1; j<n; j++)
            return i*j;
    }
}
```

A. $O(m)$
B. $O(n^2)$
C. $O(m+n^2)$
D. $O(1)$

Question 3 (0.8 Mark)

What are the articulation points (i.e., cutting vertices) in the graph below (More than one answer may be selected)?



A. `A`
B. `B`
C. `C`
D. `D`

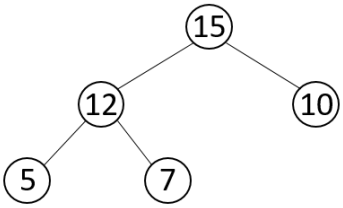
Question 4 (0.8 Mark)

Which data structures store and retrieve elements by `key`s (More than one answer may be selected)?

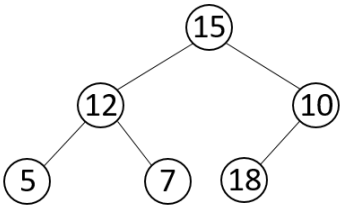
A. `std::tuple`
B. `std::map`
C. `std::set`
D. `std::pair`

Question 5 (0.8 Mark)

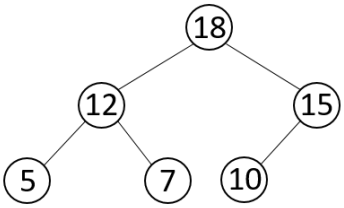
Given the binary max-heap below, what will the tree structure look like after a new element, `18`, is inserted into it?



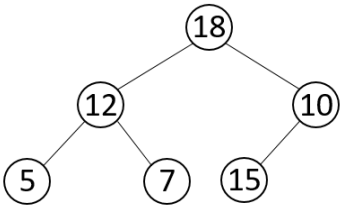
A.



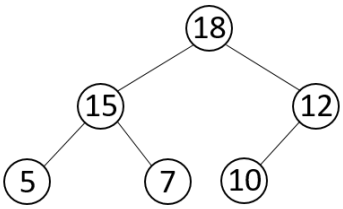
B.



C.



D.



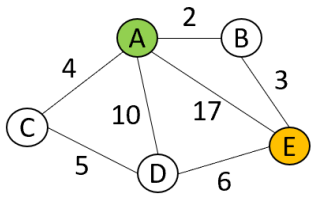
Question 6 (0.8 Mark)

Which of the following is TRUE about `template`, `std::iterator`, `reference`, and `pointer`?

A. We can define more than one `template` variable for a class or a function.
B. Suppose `iter` is an `std::iterator` variable, then we can use `iter++;` or `(*iter)+=1;` to move to the next element within a container.
C. A `reference` variable is an alias for an already existing variable. It can be used in the same way as how you use a `pointer` to the variable.
D. All of `template`, `std::iterator`, `reference`, and `pointer` store the memory address of things.

Question 7 (0.8 Mark)

Suppose you are using Dijkstra's algorithm to find the shortest path from `A` to `E`. Which of the following statement is/are TRUE about the distances of nodes as calculated by the algorithm (More than one answer may be selected)? 'Distance' of a node is the weight of the current path from `A` to the node.



A. The distance of `A` is always `0` and will not be updated by the algorithm.
B. The distance of `E` should not no longer be positive infinity once the algorithm updates the distance of `E`.
C. The distance of `D` is initialised to positive infinity by the algorithm.
D. The algorithm terminates immediately once the distance of E is updated.

Question 8 (0.8 Mark)

The visiting orders of a Preorder traversal and an Inorder traversal of a tree are shown below:
Pre-order: `A B D H E C F I G J K`
In-order: `D H B E A I F C J G K`
So what is the visiting order a Postorder traversal of the same tree?

A. `H D E B I F J K G C A`
B. `H D E B F I J K G C A`
C. `H D E B I F J K C G A`
D. None of the above.

Question 9 (0.8 Mark)

What is the worst-case complexity of removing an element from a binary heap? Suppose n is the number of elements in the heap. You may consider either a max-heap or a min-heap; that will not affect the result.

A. $O(\log n)$
B. $O(n)$
C. $O(n \log n)$
D. $O(n^2)$

Question 10 (0.8 Mark)

Which of the following is/are NOT true about adjacency_matrix and adjacency_list?

A. Suppose a graph has `n` vertices. The size of the adjacency_matrix is fixed to n^2; the size of the adajcency_list depends on the number of edges in the graph.
B. Suppose a graph has `m` edges (no self-loops) and is undirected. The number of `true`s or `1`s in the adjacency_matrix would be $2m$.
C. Suppose a graph contains the maximum number of edges it could possibly have. The adajcency_list may take more memory space than the adajcency_matrix used to represent the graph.
D. Given a vertex in a directed graph, the degree of this vertex may not equal the sum of the in_degree and out_degree of the vertex.

Question 11 (0.8 Mark)

What is the time complexity of inserting a new element before the fourth element in a singly linked list? Suppose the length of the linked list is n, which is bigger than `4`.

A. $O(1)$
B. $O(n)$
C. $O(n \log n)$
D. $O(n^2)$

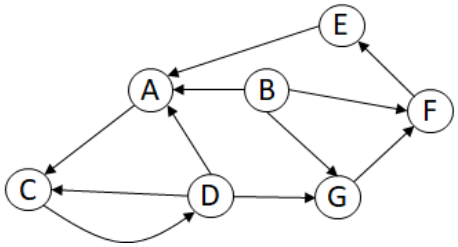
Question 12 (0.8 Mark)

Which of the data structures below can be iterated over using `std::iterator`? (More than one answer may be selected)? To iterate over a data structure means to access all the elements in the data structure one by one.

A. `std::tuple`
B. `std::map`
C. `std::set`
D. `std::list`

Question 13 (0.8 Mark)

Given the directed graph below, how many vertices are there in the largest strongly connected component (SCC) in the graph?



A. `7`
B. `6`
C. `3`
D. `4`

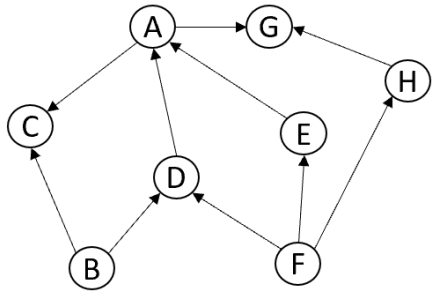
Question 14 (0.8 Mark)

Suppose you use the simple division hash function, `h(k)=k%n` and quadratic probing to insert a sequence of elements: `4,5,8,10,` one by one to an array (Suppose the length of array: n=6). What would be the array position (or index) to store `10`?

A. `0`
B. `1`
C. `3`
D. `4`

Question 15 (0.8 Mark)

Which of the following is/are NOT the result of topological sorting on the graph below (More than one answer may be selected)?



A. `A, B, F, D, E, H, G, C`
B. `B, C, F, D, E, A, H, G`
C. `F, H, E, B, D, A, C, G`
D. `B, F, D, E, H, A, G, C`

Question 16 (3 Marks)

Considering a collection of elements `{3,7,6,2,9,4}`, what insertion order of these elements would produce a Binary Search Tree (BST) that has the biggest height?

Please give at least four insertion orders that meet the above requirement (word limit: 100).

Question 17 (3 Marks)

Can you briefly describe what the function below does in plain English (word limit: 100)?

```
int fun(int n){
    std::queue<int> q;
    q.push(0);
    q.push(1);

    int a, b;
    for (int i = 0; i < n; i++) {
        a = q.front();
        q.pop();
        b = q.front();
        q.pop();
        q.push(b);
        q.push(a + b);
    }
    std::cout << a << std::endl;
}
```

Question 18 (3 Marks)

Suppose we wish to store `80` non-negative `int`s in a two-dimensional array (i.e., a square matrix like `int A[10][10]`). Can you design a mechanism to insert these `int`s into the array so that we can check if an `int` exists in this two-dimensional array efficiently? This mechanism should perform better than the exhaustive approach (i.e., iterating over the whole matrix), which has the time complexity of $O(n^2)$.

Please briefly describe your method in plain English, pseudo-code, or C++ code (word limit: 200).

Question 19 (3 Marks)

Suppose we have a square matrix: `int matrix[n][n]`, where `n>0`. Can you briefly describe what the code below does (word limit: 100)? You may draw a figure to illustrate your idea if you want.

```
for (int x = n-1; x > -1; x--) {
    for (int y = n-1; y > -1; y--) {
        if (x < n-1 && y < n-1) {
            matrix[x][y] += std::min(matrix[x+1][y], matrix[x][y+1]);
        } else if (x < n-1) {
            matrix[x][y] += matrix[x+1][y];
        } else if (y < n-1) {
            matrix[x][y] += matrix[x][y+1];
        }
    }
}
std::cout << matrix[0][0] << std::endl;
```

Question 20 (3 Marks)

Suppose we have a set of numbers `{1,8,5,2,6,3,9}`, and we wish to find a subset of these numbers so that the sum of these numbers is as close to 20 as possible yet still no greater than 20.

For example,

- the subset `{1, 8, 5}` is a feasible solution, which gives the sum of 14;
- the subset `{1, 8, 5, 2}` is a better solution because it has a greater sum of 16;
- the subset `{1, 8, 5, 2, 6}` is not a solution because the sum 22 is greater than 20.

Below shows an algorithm to find such a subset. Can you tell what strategy the code takes? Does this code guarantee to get the optimal result? (word limit: 100)

```
std::vector<int> v = {1, 8, 5, 2, 6, 3, 9};
std::sort(v.begin(), v.end());
int sum = 0, sum_limit = 20;
for (int i = 0; i < v.size(); i++) {
    if (sum + v[i] <= sum_limit) {
        sum += v[i];
    }
}
std::cout << sum << std::endl;
```

Question 21 (3 Marks)

Can you describe an algorithm that can obtain an optimal subset of the numbers for Question 20? The optimal subset should achieve the largest sum (no greater than 20).

You can explain in plain English, pseudo-code, or C++ code (word limit: 200).