The University of Melbourne
Department of Computing and Information Systems

COMP20007

Design of Algorithms
June Assessment, 2016

Student Number:

Identical Examination papers: None.

Exam Duration: Three hours.

Reading Time: Fifteen minutes.

Open/Closed Book: Closed Book.

Length: This paper has 22 pages including this cover page.

Total Marks: 85

Authorized Materials: None.

Instructions to Invigilators: Students will write all of their answers on this examination paper. Students may not remove any part of the examination paper from the examination room.

Instructions to Students: This paper counts for 60% of your final grade. All questions should be answered in the spaces provided on the examination paper. You may make rough notes, and prepare draft answers, on the blank pages, and then copy them neatly into the boxes provided. You are not required to write comments in any of your code fragments or functions.

Throughout you should assume a RAM model of computation where input items fit in a word of memory, and basic operations such as $+ - \times /$ and memory access are all constant time.

Calculators: Calculators are not permitted.

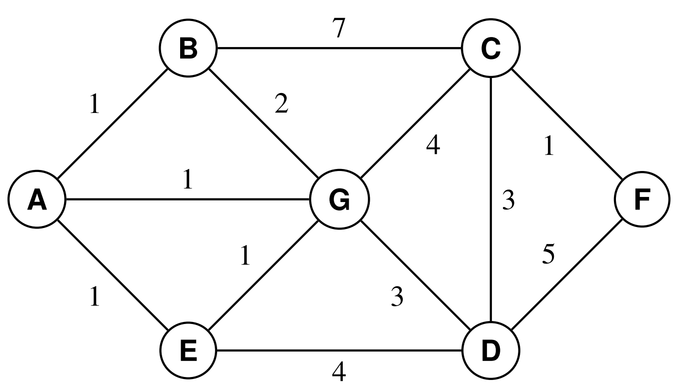
Library: This paper may be held by the Baillieu Library.

Question 1 (7 marks).

(a) (1 mark) How many edges are in a Minimum Spanning Tree of a graph with n nodes?

(b) (3 marks) Consider the graph shown below. What order are edges added into a Minimum Spanning Tree using Kruskal's algorithm on this graph? Specify an edge using a pair of node labels in alphabetical order (e.g. write the edge between G and D as DG, not GD). If there are ties, prefer the edge with the lowest alphabetical order (e.g. AG $<$ BD).

(c) (3 marks) What order are nodes added into a Minimum Spanning Tree using Prim's algorithm beginning at node G on the graph? If there are ties, prefer the node with the lowest alphabetical order (e.g. A $<$ G).



Question 2 (10 marks).

A hashtable structure is defined as follows:

```
/* Hash functions take the key, the table size */
typedef unsigned int (*Hash)(int e, unsigned int size)

struct hash_table_t {
    /* Number of spaces in the table*/
    unsigned int size;

    /* An array of pointers to int/
    /* The table[i] is NULL if bucket i is empty */
    int **table;

    /* Hash function */
    Hash hash1;

    /* Second hash function for double hashing */
    Hash hash2;
}

typedef struct hash_table_t *HT;
```

Insertion by double hashing is implemented like this:

```
void hash_insert(HT ht, int e) {
    unsigned int hash = ht->hash1(e, ht->size);
    unsigned int i = 0, h=hash;

    while (ht->table[h] && i < ht->size) {
        h = (hash + i * ht->hash2(e, ht->size)) % ht->size;
        i++;
    }
```

if (/* Check that we found an empty space */) **/* Line A */**

/* Insert element e */ **/* Line B */**

```
    else
        fprintf(stderr, "Aborting insertion: table full\n");
```

(a) (2 marks) Complete Line A to check that there is space in the table.

(b) (2 marks) Complete Line B (using as many lines as you need) to insert element e.

(c) (3 marks) Now consider the second hash function for double-hashing.

```
unsigned int hash2(int e, unsigned int size) {
       /* Line C */
}
```

Suppose we simply want `hash_insert` to use linear probing. Fill in Line C so that `hash2` does the right thing.

(d) (3 marks) Suppose instead that we implement double hashing, so `hash2` is

```
unsigned int hash2(int e, unsigned int size) {
       return e % 3 + 2;
}
```

List the conditions `size` must satisfy to make this a good second hash function. Explain why.

This page intentionally (almost) blank.

Question 3 (7 marks).

A cryptographic hash function H takes a message m and outputs a digest d. That is,

$$H(m) = d.$$

The hash function itself is completely public—anyone can compute $H(m)$ for any values of m they like. The output always has the same length (typically 256 bits).

Complete the following definitions:

(a) (1 mark) H is *collision-resistant* if it is computationally infeasible (meaning too hard) to find x, y such that $x \neq y$ and

(b) (2 marks) H is *preimage-resistant* if, given D, it is computationally infeasible (meaning too hard) to find

(c) (2 marks) Prove that collision-resistance implies preimage-resistance.

(d) (2 marks) When computing a digital signature on some message m, we usually hash first to get $D = H(m)$, then compute the signature on D. Suppose H is not collision-resistant. Explain how an attacker could forge a signature. (It doesn't have to be a signature on a meaningful message.)

Question 4 (9 marks).

Establish tight upper bounds on the worst case time complexity for the three operations that are defined on disjoint sets, for each of the three data structures. You should assume that the disjoint set contains n items at the time of the operation. You should also assume that index lookup and key comparison require $O(1)$ time.

In each cell, please omit the $O()$ and simply write 1, $\log n$, n, $n \log n$, or n^2.

	`makeset(`x`)`	`find(`x`)`	`union(`x`, `y`)`
singly-linked lists			
singly-linked lists; each cell has a pointer to the head of its list; an index maps elements to their locations in the list structure			
directed trees with union-by-rank (no path compression)			

This page intentionally (almost) blank.

Question 5 (8 marks).

Insertion Sort is a well known greedy sorting algorithm that repeatedly removes the next element from the input, storing it in sorted order. Consider the following C code for sorting integers (with line numbers added).

```
void isort(int *A, int n) {
    for(int i = 1; i < n; ++i) {
        int temp = A[i];
        int j = i;
        while(j > 0 && temp < A[j - 1]) {
            A[j] = A[j - 1];
            --j;
        }
        A[j] = temp;
    }
}
```

(a) (4 marks) What is the big-O running time of `isort` in terms of `n`? Justify your answer.

(b) (4 marks) You decide to extend the above implementation so that it supports sorting of strings. For this you need a string comparison function `int compare(char *a, char *b)` which returns -1 if `a` $<$ `b`, 0 if `a` $=$ `b` and 1 if `a` $>$ `b`. Provide an efficient C implementation for this function.

Question 6 (8 marks).

Solve the following recurrence relations and write an expression in the form of $O()$ (2 marks each)

(a) $T(n) = 8T(n/2) + n^3$

(b) $T(n) = T(n/2) + n^3$

(c) $T(n) = T(n-1) + n^c$ where c is a positive integer

(d) $T(n) = 2T(n-1) + 1$

This page intentionally (almost) blank.

Question 7 (6 marks).

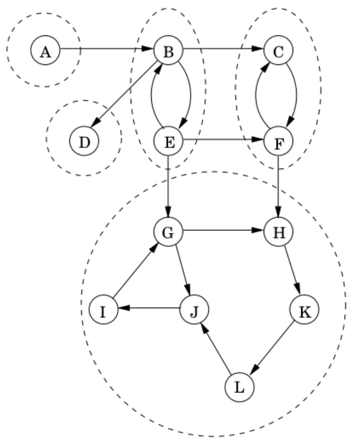
(a) (3 marks) Draw a graph, labelling the vertices with letters beginning at `A`, so that the queue in a Breath First Search of the graph beginning at vertex `A` would have a maximum length equal to 4.

(b) (3 marks) Draw a graph, labelling the vertices with letters beginning at `A`, where the stack in a Depth First Search of the graph beginning at vertex `A` would have a maximum length equal to 4.

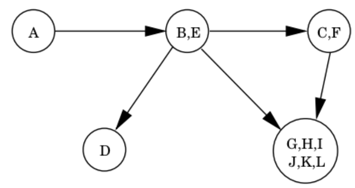
Question 8 (7 marks).

Consider the graphs shown in (a) and (b) below (from Dasgupta et al), and the table of graph functions.

(a)



(b)



Function	Description
`clear()`	sets all nodes to be ‘unvisited’
`delete_subgraph(nodes)`	delete the nodes and incident edges
`empty()`	the graph is empty
`dfs()`	perform DFS traversal and label all nodes with their pre- and post-numbers
`list_edges()`	return a list of edges
`list_nodes()`	return a list of nodes
`list_visited()`	return a list of visited nodes
`list_unvisited()`	return a list of unvisited nodes
`reverse()`	modify the graph by reversing the directions of its edges
`find_min_pre()`	return the node having the minimum pre number
`find_min_post()`	return the node having the minimum post number
`find_max_pre()`	return the node having the maximum pre number
`find_max_post()`	return the node having the maximum post number
`print(nodes)`	print a space-separated list of nodes followed by a newline
`visit(n)`	find all nodes reachable from n and mark them visited

(a) (1 mark) What is the name standardly given to the subgraphs of (a) that are indicated using dashed circles and ovals?

(b) (1 marks) What is the efficiency of the standard algorithm for Depth First Search, expressed using big-O notation.

(c) (5 marks) Suppose you have access to the functions defined in the above table, all of which operate on a global graph. Suppose that a graph has already been loaded into memory. Write pseudocode to identify these subgraphs and print their nodes. The order of nodes on a line, and the order of lines, is not important. For example, the expected output for the graph in (a) above would be the nodes shown in (b), formatted as follows:

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

Question 9 (5 marks).

Suppose a text document has 1000 characters with the following rates of occurrence:

Character	A	B	C	D	E	F	G
	500	250	100	45	45	30	30

(a) (2 marks) Give the Shannon-Fano code for this data. You may draw the tree.

(b) (2 marks) Give the Huffman code for this data. You may draw the tree.

(c) (1 mark) Give an expression for the minimum possible average number of bits that could be used to encode characters occurring with this frequency distribution.

Question 10 (10 marks).

Two integers are *coprime* if they don't have any common factors (except 1). *Euler's totient function* $\Phi(N)$ is the number of numbers between 1 and N that are coprime with N. Euler's Theorem says that, for all N and for all m coprime with N,

$$m^{\Phi(N)} = 1 \bmod N.$$

You generate an RSA public key by selecting two large primes p and q, then setting $N = pq$, and you generate a public encryption key e.

(a) (1 mark) Fill in the box:

Given N and e, you need a decryption key d such that

$de = \boxed{} \bmod \Phi(N)$.

(b) (2 marks) Rewrite the equation from part (a) without using mod.

(c) (2 marks) Using Euler's theorem, show that if m is coprime with N then

$$(m^e)^d = m \bmod N.$$

(d) (1 mark) In practice, we pad m with some randomness before encrypting. Why?

(e) (2 marks) Fill in the gaps in the following code for the extended Euclid algorithm

function extended-Euclid(a,b)

Input: two postitive integers a and b with $a \geq b \geq 0$.

Output: Integers x, y, g such that $g = \gcd(a, b)$ and $ax + by = g$.

```
if b=0:  return(1,0,a)
(x', y', g) =                    /* LINE A: Fill in */
return                 /* LINE B: Fill in */
```

Line A should be filled in with:

Line B should be filled in with:

(f) (2 marks) Suppose $N = 33$. This is the product of two small primes. Then $\Phi(N) = (p-1)(q-1) = 20$. Suppose your public encryption key is $e = 3$. Compute the private decryption key d.

Question 11 (8 marks).

This problem explores the complexity classes P, NP and NP-complete.

Suppose you can buy a special box $\mathcal{B}$ that solves any NP-complete problem that was discussed in lectures or in tutorial problems (SAT, Knapsack, Travelling salesman, *etc.*). You have to choose a particular problem before you buy $\mathcal{B}$.

You want to solve PARTITION: Given a set $a_1, a_2, a_3, \ldots, a_n$ of integer values, divide the set into two so that the sums of each subset are as close to equal as possible. More formally, find a subset I of $\{1, 2, \ldots, n\}$ that minimises

$$D = |\Sigma_{i \in I} a_i - \Sigma_{i \notin I} a_i|$$

where $|\cdot|$ denotes the absolute value.

For example, if you're given $\{1, 6, 2, 7\}$, a correct answer is $I = \{1, 4\}$, which gives the partition $\{\{1, 7\}, \{6, 2\}\}$, for which $D = 0$. If you're given $\{7, 13, 4\}$ then the best answer is $I = \{2\}$ or $I = \{1, 3\}$. This implies the partition $\{\{7, 4\}, \{13\}\}$, for which $D = 2$, which is the minimum possible.

You will show how to use $\mathcal{B}$ to solve PARTITION.

(a) (6 marks) Say which NP-complete problem you would like $\mathcal{B}$ to solve. Then describe how to use $\mathcal{B}$ to solve PARTITION. Justify your answer.

(b) (1 mark) Suppose someone finds a polynomial-time solution for the problem you used $\mathcal{B}$ to solve. What, if anything, can you infer about the complexity of PARTITION based on your answer to part (a)?

(c) (1 mark) Suppose someone proves that the problem you used $\mathcal{B}$ to solve cannot be solved in polynomial time. What, if anything, can you infer about the complexity of PARTITION based on your answer to part (a)?

Overflow Answers

The boxes here are for emergency use only. If you do need to use this page, indicate **CLEARLY** in your previous answer that you have continued onto this page. Without such an indication, it is possible that this part of your answer will be overlooked.