Xi'an Jiaotong-Liverpool University

西交利物浦大学

Paper CODE	EXAMINER	DEPARTMENT	TEL
CSE 204		Computer Science and Software Engineering	

2nd SEMESTER 2018/19 FINAL EXAMINATION

Undergraduate – Year 3

Complexity of Algorithms

TIME ALLOWED: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This is a closed-book examination, which is to be written without books or notes.
2. Total marks available are 100. This accounts for 80% of the final mark.
3. The number in the column on the right indicates the marks for each question.
4. Answers should be written in the answer booklet(s) provided.
5. Only solutions in English are accepted.
6. All materials must be returned to the exam supervisor upon completion of the exam. Failure to do so will be deemed academic misconduct and will be dealt with accordingly.

Notes:

- **To obtain full marks for each question, relevant and clear steps should be included in the answers.**
- **Partial marks may be awarded depending on the degree of completeness and clarity.**

Question 1: Algorithm Analysis [30 marks]

a) What is the asymptotic value of the expression $\sum_{i=1}^{n} i^2$ as a function of n by using the big-O notation? [3 marks]

b) Given two 64-bit integers a, n, here is an algorithm to compute a^n:

```
Power(a,n):
1. If n = 0: return 1.
2. If n = 1: return a.
3. If n is even:
4. Return Power(a×a,n/2).
5. else:
6. Return a×Power(a×a,(n−1)/2).
```

Assume throughout this problem that we do not need to worry about overflow (a^n fits into a 64-bit integer variable) and that each operation on a 64-bit integer takes O(1) time.

i. Let A(n) denote the running time of Power(a,n). Write a recurrence relation for A(n). [2 marks]

ii. What is the solution to your recurrence A(n)? Use $\Theta(\cdot)$ notation. [3 marks]

c) Consider the pseudo-codes representing two computer functions called *loopA* and *loopB* as follows:

```
loopA (array T[], integer n) {
for i from n to 2
   for j from 1 to i-1
      if T[j]>T[j+1] then
      swap T[j] and T[j+1]
      endif
   endfor
endfor
}
```

```
loopB (integer n) {
   for s from 1 to n
      for t from 1 to s
         for k from 1 to t
          a←t+k
         endfor
      endfor
   endfor
}
```

i. How many times will the *swap* function be performed in the best case and worst case of *loopA*? Justify your answer. [4 marks]

ii. Express the time complexity of the *loopB* function using Big-Oh notation. Justify your answer. [4 marks]

iii. Which program is faster? Justify your answer. [4 marks]

d) Given a binary search tree of height h, we wish to find out the value of its kth element.

i. Complete the following linear time function *findKth* (using pseudo-code), which stops after finding the kth element of the given binary search tree. [7 marks]

ii. Compute the time complexity of *findKth* function by using Big-Oh notation. [3 marks]

```
findKth(T; k)
Input: a binary search tree T (T.Left and T.Right denote the left and right subtrees,
respectively) and an integer k
Output: the value of kth element of T, T.value.
(you can use pos as a global variable, which is initialized to 0)
1______________________
2______________________
3______________________
…
```

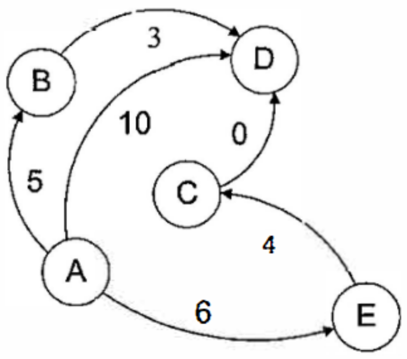
Question 2: Heap and Graph [30 marks]

a) Consider a binary heap. Print the keys as encountered in a preorder travel. Is the output sorted? Justify your answer. [5 marks]

b) Draw the binary min-heap that results from inserting: 77, 22, 9, 68, 16, 34, 13, 8 in that order into an initially empty binary min heap. You do not need to show the array representation of the heap. You are only required to draw intermediate heaps and circle the final step. [5 marks]

c) What is time complexity of the binary min-heap algorithm? [3 marks]

d) Use the following graph for this problem. Where needed and not determined by the algorithm, assume that any algorithm begins at node A.



i. Draw both the adjacency matrix and adjacency list representations of this graph. [6 marks]

ii. Step through Dijkstra's Algorithm to calculate the single source shortest path from A to every other vertex. You only need to show your final table, but you should show your steps in the table below for partial credit. Show your steps by crossing through values that are replaced by a new value. Note that the next question asks you to recall what order vertices were declared known. [5 marks]

Vertex	Known	Distance	Path
A			
B			
C			
D			
E			

iii. What is the shortest (weighted) path from A to D? [3 marks]

iv. What is the length (weighted cost) of the shortest path you listed in part iii.? [3 marks]

Question 3: Number Theory and Cryptography [15 marks]

a) Evaluate 103^{-1} mod 120. [4 marks]

b) Alice and Bob read a scientific article on the public-key RSA cryptosystem. They have decided to apply the RSA algorithm on a simple example in order to check their understanding of this cryptosystem. To this end, Alice chooses the public key ($n = 143$ and $e = 7$). Bob chooses an integer between 0 and 142, then encrypts it and sends the number 27 to Alice. Can you help Alice finding out the original integer chosen by Bob? [5 marks]

c) Modify Euclid's algorithm as follows

```
function Newclid(a,b)
if a<b: swap a and b
if b=0: return a
return Newclid(a-b,b)
```

i. Does Newclid(a,b) still compute gcd(a, b)? Briefly justify your answer. [3 marks]

ii. Show that Newclid(a,b) is exponentially slower than Euclid(a,b) on certain inputs. [3 marks]

Question 4: NP-Hardness [25 marks]

a) Define the Vertex Cover problem. [5 marks]

b) What is an α-approximation algorithm for a maximization problem. [5 marks]

c) Using reductions from Vertex Cover show that the following problem is NP-complete.

Independent Set: Given a graph G = (V, E) and an integer k. Does G contain an independent set S of cardinality at least k? [A set $S \subseteq V$ is independent (stable) if every pair of vertices in S are non-adjacent.] [8 marks]

d) Recall that there exists a dynamic-programming algorithm for the Knapsack problem that has a running time of O(nW), where n is the number of items, and W is the knapsack's capacity. However, the Knapsack problem is also known to be NP-hard. Therefore, given any problem in NP, we can reduce it to Knapsack in polynomial time, and then solve the Knapsack instance with the dynamic-programming algorithm. This would give us a polynomial-time algorithm for every problem in NP, and it would therefore imply that P=NP. What is the fallacy/mistake in this argument? Justify your answer. [7 marks]

END OF EXAM PAPER