May 2015

Coventry University

Faculty of Engineering & Computing

210CT
Programming, Algorithms and Data Structures

Instructions to candidates:

Time Allowed: **2 hours**

This is a **Closed Book** examination
Answer: **All** questions

Solution: Answers and marking guide.
DO NOT COPY.

The total number of qestions in this paper: 4

Start each question on a new page and carefully identify your answers with the correct question number

For this examination you will be supplied with the following: 1 Answer Book

---

You may take this question paper away at the end of the examination.
Please keep it in a safe place for future reference.

---

1. Total for Question 1: 20

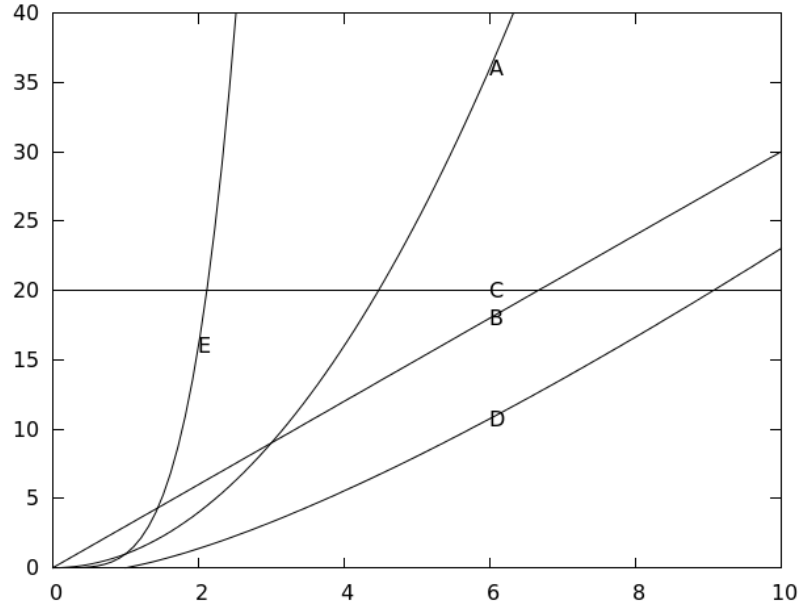
(a) Write the functions below in order of asymptotic growth rate, beginning with the largest. If any of the functions have the same growth rate, be sure to show it. (10)

- $4n^2$
- n^4
- $n + n$
- $n^{1.5}$
- $3n.log(n)$
- $40n$
- $log(n)$

Solution:

- $n^4, 4n^2, n^{1.5}, 3n.log(n), [n + n, 40n], log(n)$
- 4 marks for at east 4 in the correct order, 8 marks for all of them without grouping the same rates and full marks for the complete correct answer

(b) Examine the graph below. The plots A, B, C, D, E represent the growth rates $n.log(n), n^2, n^4, 20, 3n$. State the function for each plot. (10)



Solution: 2 marks each. . .

A n^2

B 3n

C 20

D n.log(x)

E x^4

Continue

2. Total for Question 2: 25

(a) Use pseudocode to describe an algorithm that finds the smallest number in a sequence. (5)

Solution: Example,

```
FIND SMALLEST(seq)
    if size(seq)<1
        raise error
    smallest←seq[0]
    for all numbers i in seq
        if i<smallest
            smallest←i
    return smallest
end
```

Marks are lost for not checking sequence length or failing to clearly show nesting of structures.

At most 2 marks for an algorithm thar doesnt work due to some oversight or error, but shows a correct intention.

(b) Write a function in the programming language of your choice that takes two sequences of numbers as parameters and returns a sequence that contains the non-unique numbers from each input. That is, those that are in the first and the second lists. There should be no duplicates in the output. You **may** use any library or language feature that can test for the presence of a value in a sequence, such as Python's `in` or `find`, but it is not a requirement. For example, given the sequences $[1, 2, 3, 4, 5]$ and $[4, 5, 6, 7]$, the answer is $[4, 5]$. (10)

Solution: An example:

```
def non_unique(a,b):
    result=[]
    for i in a:
        if i in b and not i in result:
            result.append(i)
    return result

if __name__ == '__main__':
    a=[1,2,3,4,5,7,7,9,9,9]
    b=[4,5,6,7]
    print non_unique(a,b)
```

- Lose 4 marks if the answer fails to check for a number

appearing in one list but multiple times.

- Lose 2 marks for not having a function
- Max of 5 marks for a partial answer

(c) i. Examine the pseudocode below and describe the effect it has on the given input sequence. (5)

```
REARRANGE(A)
n ← length(A)
    for j ← 0 to n
        for i ← 0 to n-1
            if (A[i] < A[i+1])
                swap( A[i], A[i+1] )
```

ii. Now show an implementation of the function in the language of your choice. You will not be penalised for small syntactical errors. (5)

Solution:

1. Sequence becomes sorted (3 marks) from high to low (2). Give up to 3 marks for describing the operations instead of the final result.
2. Anything suitable... a function (1) that has two nested loops (2) that sorts the data (3).

Continue

3. Total for Question 3: 25

To sort $A[p \ldots r]$ using the Quick Sort algorithm:

- Select a pivot, P
- create a less-than (L) and greater-than (G) list
- For each value in A, place it in L if it is less than P or G if it is greater (or equal)
- Quick Sort L and G
- Return L+P+G (where + is concatenation)

(a) Write the pseudocode for the Quick Sort algorithm. You can assume that the + symbol can be used to concatenate lists. (10)

Solution:

```
QUICK-SORT(l):
    if length(l) <= 1
        return l
    L←[]
    G←[]
    P=median(l[0],l[length(l)-1],l[length(l)/2])
    for each i in l
        if i<P
            L.append(i)
        else
            G.append(i)
    return L+P+G
```

Marks are lost for poor checking of size in base-case, poor pivot selection, incorrect partitioning, incorrect reforming, etc.

(b) Use big-O notation to describe the run-time performance of the quick sort algorithm, relative to input size, n, assuming the merge operation is $O(n)$. (5)

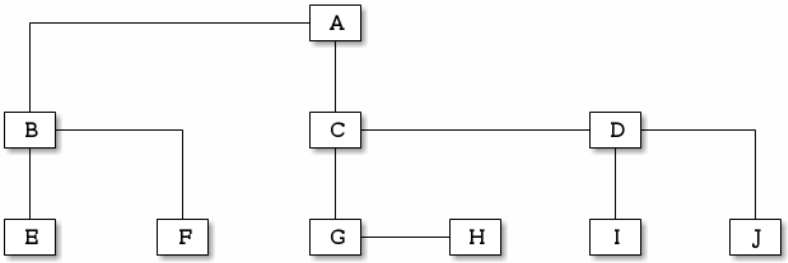
Solution: (average case) Division occurs $\log(n)$ times and the scanning/concatenation for each requires $O(n)$ time, so $O(n.\log(n))$

(c) In the language of your choice, show an implementation of the quick sort algorithm. It does not matter if the sort is in-place, returns a new sequence or takes a pointer to an empty list as a parameter. (10)

Solution: Full marks for a reasonable implementation. Marks are lost for logical errors, not syntactical ones, although consistency is important in whichever syntax is chosen.

4. Total for Question 4: 30

(a) Describe how a **breadth first search** (BFS) is performed on a graph and state the order in which the nodes of the graph below will be visited, beginning with **A**. Assume that visited nodes will be remembered and not visited again, and that left-hand edges are selected first. (10)



Solution: Up to 6 marks for a reasonable description that is detailed enough to follow (children examined in order, recursing only when all children of a given node are done)

Remaining 4 marks for the correct sequence: A B C E F G D H I J (lose 1 mark per mistake)

(b) Explain the effect of a small change in the input to a **good** hashing algorithm has on its output. Use a real-world example to help explain your answer. (5)

Solution: 1 mark for explaining that a small change in input leads to a large change in output, another for stating that the change in output cannot be inferred from the change in input. Final 3 marks are for an example, such as password storage or checksums.

(c) i. In a binary tree, what is the maximum and minimum number of children a node may have? (2)

ii. Draw the binary tree resulting in the insertion of the numbers 15,4,7,10,5,2,3,1. (3)

iii. Ideally, a binary tree should be "balanced". Explain what this means and describe a worst-case input sequence that leads to an unbalanced tree. (3)

iv. Explain how a binary tree can be stored, maintaining all relationships, in an array. Be sure to explain how to find the location of a node's children and parent given its index. (7)

Solution:

1. Maximum 2 children, minimum 0 (1 mark each)
2. See output of array, below
3. A balanced tree has roughly equal (± 1) numbers of nodes in the trees headed by its children (1 mark). Order of input effects the tree structure and sorted input (1 mark) will result in a tree with all nodes having only one child (1 mark).
4. First item at index 0 (1 point). Children at 1 and 2. For any node at i, children (left, right) are at $2i + 1, 2i + 2$ (3 points) and the parent is at $\lfloor \frac{i-1}{2} \rfloor$ (3 points)

```
#Assumes list is in which empty cells contain -1
def insert(n, l, p=0):
    while p>len(l)-2:
        #Double in size when needing to add more
        l.extend([-1]*len(l))
    if l[p]==-1:
        l[p]=n
    else:
        if n<=l[p]:
            insert(n,l,2*p+1)
        else:
            insert(n,l,2*p+2)

input=[15,4,7,10,5,2,3,1]
tree=[-1]*len(input)

for i in input:
    insert(i,tree)
print tree
[15, 4, -1, 2, 7, -1, -1, 1, 3, 5, 10, -1, -1, -1, -1, -1]
```

END