Ordinary Exam
DATA STRUCTURES AND ALGORITHMS DVG B03

130115 08:15 – 13:15

Course Coordinator: Donald F. Ross

Help Material: A Dictionary from the student's home language to English.

***** OBS *****

Students who have studied the course as from (>=) Autumn Term 2006

Grading Levels:

Course:	Max 60p, pass with special distinction 50p, pass with distinction 40p, pass 30p **(of which a minimum 15p from the exam, 15p from the labs)**
Exam:	Max 30p, grade 5: 26p-30p, grade 4: 21p-25p, grade 3: 15p-20p
Labs:	Max 30p, grade 5: 26p-30p, grade 4: 21p-25p, grade 3: 15p-20p

Students who have studied the course before (<) Autumn Term 2006

Grading Levels:

Course:	Max 60p, pass with special distinction 50p, pass with distinction 40p, pass 30p **(of which a minimum 20p from the exam, 10p from the labs)**
Exam:	Max 40p, grade 5: 34p-40p, grade 4: 27p-33p, grade 3: 20p-26p
Labs:	Max 20p, grade 5: 18p-20p, grade 4: 14p-17p, grade 3: 10p-13p

Write legibly – read all questions carefully

(1) Give a short answer to each of the following questions:-

(a) Show the relationship between a binary search tree, a complete tree and a full tree with the help of a Venn-diagram from set theory.
(b) What is the maximum load factor in hashing with quadratic probing as the collision management technique which guarantees that a space can always be found?
(c) What is ”big-O”?
(d) What does Warshall’s algorithm do?
(e) Give a definition of an AVL-tree.
(f) Which grammatical class in natural language corresponds to a relation in the Entity Relationship model?
(g) Which algorithm requires a DAG as a starting point?
(h) Which data structures are ordered?
(i) Which algorithm is $O(n^3)$?
(j) Give a definition of a recursive function.

Total 5p

(2) Abstraction

Discuss in detail why abstraction is so important in data structures. Are there any possible disadvantages to abstraction? Which 3 definitions of the term abstraction have we used in this course?

5p

(3) Sequence

(a) Give a recursive definition of a sequence.

(1p)

(b) From your definition in (a) above, write recursive pseudo code to count up the number of elements in a sequence
State all assumptions.
Give an example of how your code works.

(2p)

(c) From your definition in (a) above, write recursive pseudo code to add an element to a sequence in increasing (sorted) order. Assume that a function called Cons which adds an element to the head of the list already exists.
Cons: element x list ➔ list.
State all assumptions.
Give an example of how your code works.

(2p)

Total 5p

(4) <u>General questions – give a detailed answer with an example</u>

(a) What is double hashing?
(b) What is a stack used for?
(c) How is a general tree transformed to a binary tree?
(d) How does the add function work in a heap?
(e) Describe a solution to the TSP-problem. TSP = Travelling Salesman Problem.

Total 5p

(5) <u>Graph – Dijkstra + SPT</u>

<u>Extend Dijkstra's algorithm</u> below in order to save and show the SPT.
(SPT: Shortest Path Tree).

```
Dijkstra ( )
{       S = {a}

        for ( i in 2..n) D[i] = C[a, i]          -- initialise D

        for (i in 1..(n-1)) {
                choose w in V - S such that D[w] is a minimum
                S = S + {w}
                foreach ( v in V-S) D[v] = min(D[v], D[w]+C[w,v])
                }
        }
```

(3p)

Apply your **<u>extended version of the algorithm</u>** to the following **<u>directed graph</u>**:-

(a, b, 13), (a, d, 12), (a, e, 20), (b, c, 60), (c, e, 50), (d, c, 30), (d, e, 40)

Start at node "a".
<u>Show each step in your calculation.</u>
<u>State all assumptions and show all calculations and intermediate results.</u>
Draw **<u>each step</u>** in the construction of the STP– i.e. show the nodes and which edges are added and subsequently removed.

(2p)

Total 5p

(6) Graph – Prim's algorithm

Apply **the version of Prim's algorithm given below** to the undirected graph:-

(a-6-b, a-3-c, a-7-d, b-1-c, b-12-e, c-4-d, c-5-e, c-9-f, d-3-f, e-8-f).

Start with node "a".

State all assumptions and show all calculations and intermediate results.

(3p)

Explain **the principles** behind Prim's algorithm.
Use the example above in your explanation.

(2p)

```
Prim ( node v) -- v is the start node
{   U = {v}; for i in (V-U) { low-cost[i] = C[v,i]; closest[i] = v; }

    while (!is_empty (V-U) ) {
        i = first(V-U); min = low-cost[i]; k = i;
        for j in (V-U-k) if (low-cost[j] < min) {min = low-cost[j]; k = j; }
        display(k, closest[k]);
        U = U + k
        for j in (V-U) if ( C[k,j] < low-cost[j] ) )   {low-cost[j] = C[k,j]; closest[j] = k; }
        }
    }
```

Total 5p