

UNIVERSITY OF CANTERBURY

# Mid-Year Examinations 2007

Prescription Number(s): COSC329

Paper Title: Algorithms and Artificial Intelligence

Time Allowed: THREE hours

Number of pages: 6

1. Answer *all* questions.
2. This exam is worth a total of 100 marks.
3. Open book examination. Calculators permitted.
4. Check carefully the number of marks allocated to each question. This suggests the degree of detail required in each answer, and the amount of time you should spend on the question.
5. Use the separate *Answer Booklet* for answering *all* questions.

**Question 1 [20 marks for the whole question]**

Examples of mixed-radix sequences are generated in lexico-graphic order and Gray code order as follows:

Lexicographic	Gray
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 1
0 1 1	0 1 0
0 2 0	0 2 0
0 2 1	0 2 1
1 0 0	1 2 1
1 0 1	1 2 0
1 1 0	1 1 0
1 1 1	1 1 1
1 2 0	1 0 1
1 2 1	1 0 0

In this example, the radices for the three positions are given by (2, 3, 2). We call this a radix vector. The radix for the middle position is ternary, while those for the leftmost and rightmost are binary. A recursive algorithm for lexico-graphic order is given below where numerical values are hard coded.

```
int a[10], r[10];
int i, j ,k, n;
void out(){
    int i;
    for(i=1;i<=n;i++)printf("%d ", a[i]);
    printf("\n");
}
void nest(int k){
    int i;
    if(k<=n){
        for(i=0;i<r[k];i++){
            a[k]=i; nest(k+1);
        }
    }else out();
}
main(){
    n=3; r[1]=2; r[2]=3; r[3]=2;
    nest(1);
}
```

- [5 marks] List up mixed-radix sequences for the radix vector (2, 3, 2, 2) in lexico-graphic and Gray order.
- [5 marks] Given the radix vector in array “r”, design an iterative algorithm for the sequences in lexico-graphic order.
- [5 marks] Design a recursive algorithm for the Gray code order.

- (d) [5 marks] Design an iterative algorithm for the Gray code order. This must run in  $O(1)$  worst case time per sequence.

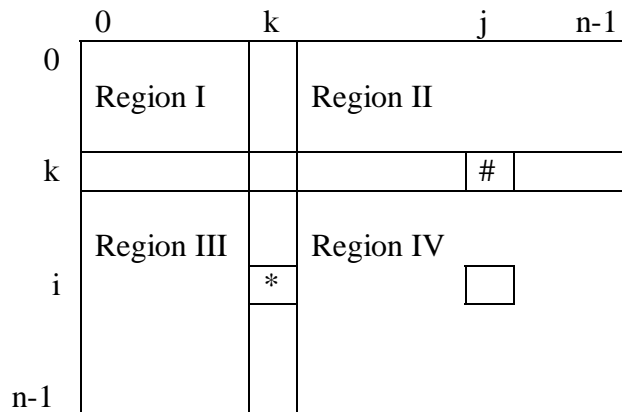
Note. Indexing for array “a” and “r” is part of design work.

**Question 2 [20 marks for the whole question]**

The following is Floyd’s algorithm for all pairs shortest paths, where array “d” is the initial distance matrix with the diagonal elements 0. The elements of “d” are updated through the computation. Vertices are numbered 0, ..., n-1.

```
for (k=0;k<=n-1;k++)
  for (i=0;i<=n-1;i++)for (j=0;j<=n-1;j++)
    d[i][j] = min(d[i][j], d[i][k]+d[k][j]);
```

We implement Floyd’s algorithm on a mesh computer. The computation is viewed as repeated update process by update axes, horizontal and vertical. A typical situation is depicted below for the k-th update. The update origin starts at point (k, k) at time 3k. The k-th row and column are horizontal and vertical update axes.



The values at \* and # meet at point (i, j) at time  $T(i, j, k)=3k+|i-k|+|j-k|$  and update the value at (i, j) if necessary. We can ensure the k-th update takes place after the (k-1)-th at every (i, j). The computation finishes at time  $T(0, 0, n-1)=5n-5$ .

The k-th update origin is fired at (k, k), and control signals go into four directions, triggering the data in the axes to go into two directions orthogonal to the axes. This is done on the whole array “d” for  $k=0, \dots, n-1$  in this order.

If the graph is acyclic in the sense that there is no edge from j to i for  $i < j$ , we can only update Region II. The following algorithm is based on this observation

```
for (k=1;k<=n-2;k++)
  for (i=0;i<=k-1;i++)for (j=k+1;j<=n-1;j++)
    d[i][j] = min(d[i][j], d[i][k]+d[k][j]);
```

- (a) [6 marks] Trace this algorithm with the following matrix at the end of each k-th iteration.

0	2	6	20
99	0	3	99
99	99	0	9
99	99	99	0

- (b) [7 marks] The mesh algorithm for Floyd started the k-th update origin at time  $3k$  at cell  $(k, k)$ . We observed the timing was not tight for Region II. If the given graph is acyclic as described above, we may be able to start update origins earlier. Based on this observation, try to design a mesh algorithm for acyclic graphs more efficient than the original mesh Floyd. For the style of description, follow that of the mesh Floyd given above. At what time can you finish the computation?
- (c) [7 marks] Trace your mesh algorithm with the above matrix at each parallel step.

**Question 3 [10 marks for the whole question]**

- (a) [5 marks] Draw a bitonic sorting network of size 16.
- (b) [5 marks] Trace your network with the input sequence

(16, 5, 6, 7, 8, 15, 1, 2, 3, 4, 14, 9, 10, 11, 13, 12)

**Question 4 [18 marks for the whole question]**

Colour	Type	Grain	Class: Suitable?
Red	Hard	Coarse	Yes
Green	Soft	Coarse	Yes
Black	Soft	Fine	No
Green	Hard	Coarse	No
Red	Soft	Coarse	No
Green	Hard	Fine	Yes
Black	Hard	Coarse	No
Red	Hard	Fine	Yes

- (a) [8 marks] The table above contains data about types of wood and their suitability for building houses. Use the ID3 algorithm to determine the attribute to branch on first. Show your working.
- (b) [3 marks] Using just this first attribute that you have determined, how well can you predict the class for (Black, Soft, Coarse)? Explain.
- (c) [5 marks] Using the nearest neighbour algorithm, determine the class of (Black, Hard, Fine) assuming that the distance function for symbolic attributes is  $d=0$  when the attribute has the same value, otherwise  $d=1$ . Show your working.
- (d) [2 marks] Consider the following function:  $f(0) = 1$ ;  $f(1) = 0$ ;  $f(2) = 1$ ;  $f(3) = 1$ . Can a single perceptron learn this function? Explain.

**Question 5 [8 marks for the whole question]**

The following is a fragment of pseudocode from an expert system that uses backward chaining to make inferences:

```
stop_car if (intersection_dangerous).  
  
intersection_dangerous if lights_red or not  
(intersection_clear).  
  
intersection_safe if not(intersection_dangerous).  
  
intersection_clear if not (vehicle_sensor).
```

- (a) [4 marks] Assuming that `lights_red` and `vehicle_sensor` are inputs to the system, rewrite the above logic for a *forward* chaining expert system.
- (b) [2 marks] Suppose you are building an expert system for detecting objects in a visual scene. There are a large number of inputs (pixels) and a large number of possible outcomes. The inputs are mostly static, i.e. they change only occasionally. Would you use forward or backward chaining to implement this system? Explain.
- (c) [2 marks] Explain why “salience” is sometimes needed in forward-chaining systems and give an example of its use. Describe an alternative way you could overcome the same problem.

**Question 6 [8 marks for the whole question]**

- (a) [5 marks] Determine the output of the SEQUITUR algorithm when run over the following string, where each input symbol is a word. Show the state of the input set  $S$  and rule set  $R$  as each new word is input.

**You say why does everything revolve around you you say  
why does everything I do confound you**

- (b) [3 marks] Suggest how SEQUITUR might be used to help build “emergent intelligence” into the world wide web.

**Question 7 [16 marks for the whole question]**

- (a) [10 marks] Use best-first search to find the shortest path from START to FINISH. Use the Manhattan distance as your cost estimate (i.e. `squares_across` + `squares_up_or_down`). Show your working.

Start	1	2		Finish
3		4		5
6		7		8
9				10
11	12	13	14	15

- (b) [3 marks] An alternative cost function might be to use the distance between a given square and the finish measured in a straight line. Would this be a better or worse estimate? Explain.
- (c) [3 marks] Are the Manhattan and straight-line distances admissible? Explain.

**END OF PAPER**