THE UNIVERSITY OF THE SOUTH PACIFIC

CS214: Design and Analysis of Algorithms
School of Computing, Information and Mathematical Sciences

Final Examination
Semester 2 2016

Mode: Face-to-Face /Blended

Duration of Exam: 3 hours + 10 minutes

Reading Time: 10 minutes

Writing Time: 3 hours

Instructions:

1. This is a closed book exam
2. This exam has four sections – I, II, III and IV
    a. Section I: 15 Marks
    b. Section II: 16 Marks
    c. Section III: 8 Marks
    d. Section IV: 6 Marks
3. All questions are compulsory.
4. Write your answers in this booklet.
5. This exam is worth 45% of your overall mark.
6. Number of pages (including this cover page): 14

Name: _____

ID: _____

Section I: True/False Questions [15 marks]

Circle either **True** or **False** in the grid provided below for Questions 1-30. Correct answers will be awarded full marks.

Question	Answer		Question	Answer		Question	Answer	
1	True	False	11	True	False	21	True	False
2	True	False	12	True	False	22	True	False
3	True	False	13	True	False	23	True	False
4	True	False	14	True	False	24	True	False
5	True	False	15	True	False	25	True	False
6	True	False	16	True	False	26	True	False
7	True	False	17	True	False	27	True	False
8	True	False	18	True	False	28	True	False
9	True	False	19	True	False	29	True	False
10	True	False	20	True	False	30	True	False

1. Preorder traversal is one way to perform a depth-first search on a tree.
2. In backtracking, if a node is visited and determined that it can possibly lead to a solution, we call the node promising.
3. A Hamiltonian Circuit can be called a tour.
4. In the 0-1 Knapsack Problem, a node is nonpromising if the total weight thus far is greater than or equal to *W*.
5. Backtracking algorithm does not depend on the set of data they are given.
6. A problem can not be solved by more than one algorithm.
7. Sequential search appears to be much more efficient than binary search.
8. The minimum number of times an algorithm will ever do its basic operation for an input size of n is called the algorithm's worst case time complexity.
9. A quadratic-time algorithm can be eventually more efficient than a linear-time algorithm.
10. Iterative algorithms execute faster than tail-recursion because no stack needs to be maintained.
11. Dynamic programming is similar to divide-and-conquer in that we find a recursive property.
12. The value of the optimal solution is computed in a bottom-up fashion.
13. Dynamic programming applies in the Travelling Salesman Problem.
14. Any spanning tree has a minimum weight.
15. Prim's algorithm always produces a minimum spanning tree.
16. In branch-and-bound problem, we are limited in a fixed way to traverse the tree.
17. Since the 0-1 Knapsack Problem is an optimization problem, we can only use branch-and-bound to solve it.
18. In best-first search and breadth-first search, items are put into a priority queue.
19. From one set of data, we can know how quickly the branch-and-bound solves the traveling salesman problem.
20. Quicksort is a partition exchange sort.

21. $O(n^2+n)=O(n^3)$
22. if $f(n)=n^2+\lg n$, then $f(n) \in \Theta(n^2)$.
23. $2^n \in \Theta(2^{n+4})$
24. $n/5 \in O(n)$
25. The worst case efficiency of the *Mergesort* algorithm that you have studied in this course (CS214) is better than *Quicksort*'s worst case.
26. The following lists (arrays) of integers will produce the worst case time complexity for *Quicksort* algorithm: "**9 8 7 6 5**" and "**5 6 8 9 10**".
27. Floyd's algorithm has an every case time complexity.
28. Dynamic Programming is a programming approach to solve problems that have a dynamic nature.
29. Floyd's Algorithm and Kruskal's algorithm can be used to solve the same problem since they both produce shortest path.
30. An instance of an optimization problem can only have one solution since the solution must have an optimal value.

Section II: Short Answer

1. Give the tree of recursive calls when using the *quicksort* algorithm provided below to sort the following list:

   103 14 169 36 130 [3 marks]

```
void quicksort (index low, index high)
{
    index pivotpoint;
    if (high > low){
        partition(low, high, pivotpoint);
        quicksort(low, pivotpoint-1);
        quicksort(pivotpoint+1, high);
    }
}
```

```
void partition (index low, index high,
                index& pivotpoint)
{
    index i, j;
    keytype pivotitem;

    pivotitem = S[low];
    j = low;
    for (i = low + 1; i <= high; i++)
        if (S[i] < pivotitem){
            j++;
            exchange S[i] and S[j];
        }
    pivotpoint = j;
    exchange S[low] and S[pivotpoint];
}
```

2. The following 5 items were given to Tom for free. Tom's Knapsack has the capacity to fit any 2 of these items. Tom can make 2 trips to take some items home. How many options Tom can have if he wants to take as many items as he can? [3 marks]

itemA: 12LB,$30; itemB: 7LB,$20; itemC: 25LB,$40; itemD: 7LB,$40; itemE: 8LB,$10

3. Find two instances of the n-Queens Problem that has no solutions. Justify your answer. [2 marks]

4. Floyd's algorithm considers two cases for shortest path from V_i to V_j using only vertices in $\{V_1, V_2, \ldots, V_{k-1}, V_k\}$ as intermediate vertices:

   Case A: do not use vertex V_k; and

   Case B: do use vertex V_k.

   a. In which case(s) the Principle of Optimality applies? Explain.

   b. Use the appropriate case(s) to establish a recursive property that can give the shortest path.

[2+2 marks]

5. Consider the following 0-1 Knapsack problem: [2+2 marks]

item 1: $20, 2kg
item 2: $30, 5kg
item 3: $35, 7kg
item 4: $12, 3kg
item 5: $3, 1kg
knapsack maximum capacity: 9kg

a. Use the Backtracking algorithm to maximize the profit. Show the state space tree.
b. What values the *maxiprofit* goes through?

Section III: Analysis of Algorithm

1. Consider the following algorithm: [1+1+1+1 marks]

```
int somefunction (int n , int Z[ ][ ]) {
    index i, j, p, q;
    for (i = 1 ; i <= n ; i++){
        for (j = 1 ; j <= n ; j++){
            for (p = 1 ; p <= n ; p++){
                for (q = 1 ; q <= n ; q++){
                    if (Z[i][j] == Z[p][q] && !( i == p && j == q)){
                        return 1 ;
                    }
                }
            }
        }
    }
    return 0 ;
}
```

a. What is the best case time complexity of the algorithm (assume n > 1)?

b. What must be true of the input data for this best case to occur?

c. What is the worst case time complexity of the algorithm?

d. What must be true of the input data for this worst case to occur?

2. Give the asymptotic running time of the *prim* algorithm below. Show that if the run time has a $O(n^5)$. [4 marks]

```
void prim (int n, const number W[][], set_of_edges& F){
   index i, vnear;
   number min;
   edge e;
   index nearest[2..n];
   number distance[2..n];

   F=∅;
   for (i = 2; i <= n; i++){
      nearest[i]=1;
      distance[i] = W[1][i];
   }

   repeat (n-1 times){
      min = ∝;
      for (i = 2; i <= n; i++)
         if (0 <= distance[i] < min){
            min = distance[i];
            vnear = i;
         }
      e = edge connecting vertices indexed by vnear and nearest[vnear];
      add e to F;
      distance[vnear] = -1;
      for (i = 2; i <= n; i++)
         if (W[i][vnear] < distance[i]){
            distance[i] = W[i][vnear];
            nearest[i] = vnear;
         }
   }
}
```

Section IV: Algorithm Design

1. Travel maps for the Company's salesmen are often changed. [2+2+2 marks]

a. To design an algorithm that can help the salesmen to find out the least-cost path on their travel maps, what data structure would you choose? Why?

b. Design an algorithm that can help the Company's salesmen to find out the number of trips between cities for a given travel map using the data structure you have chosen (in point a, above). Assume at most one trip can be made between any two cities.

c. A salesman needs to do the promotion in any 3 of the cities on his travel map. The salesman is currently in his home city. The Company requires him to begin the trip within 2 days' time. Name 2 algorithms you have learned from this course (CS214) that can be used to help the salesmen finding out about the least cost to travel to any 3 of the cities? Which algorithm is more efficient? Why?

Extra Working Page 1

Extra Working Page 2

Extra Working Page 3

THE END