NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

FINAL EXAM FOR
Semester 2, AY2013/2014

CS3230 – DESIGN AND ANALYSIS OF ALGORITHMS

26 April 2014 Time Allowed: 2 hours

Instructions to Candidates:

1. This examination paper contains **FIVE** (5) questions and comprises **TEN** (10) printed pages, including this page.
2. Answer **ALL** questions.
3. Write ALL your answers in this examination book.
4. This is an **OPEN BOOK** examination.
5. You can **freely quote** standard algorithms and data structures covered in the lectures, textbook, homework, and programming assignments. However, please explain **any modifications** you make to them.
6. Unless otherwise specified, you are expected to **prove (justify)** your results.
7. All logarithms are in base 2, unless otherwise stated.

Matriculation Number: _____

QUESTION	POSSIBLE	SCORE
Q1	10	
Q2	10	
Q3	10	
Q4	5	
Q5	5	
TOTAL	**40**	

Score: (__+__) + (__) = (___)	CS3230

Q1. (10 points)

(a). (4 points) Solve the following recurrence, giving your answer for Q(n) in Θ-notation:

$Q(n) = Q(n-1) + 5n^2$ for $n > 1$, $Q(1) = 5$.

$R(n) = 6R(n/2) + \Theta(n^2)$ for $n > 1$, $R(1) = \Theta(1)$.

(b). (6 points) Suppose that instead of expanding a table by doubling its size when the table is full, we expand it by multiplying its size by 4 when the table is full. Show that the amortized cost of TABLE-INSERT using this strategy is bounded above by a constant. What is the constant amortized cost per operation? Use any method that you like.

Q2. (TREASURE-HUNT) (10 points)

There are **N** treasure boxes in an island left by some rich king of the past. Each treasure box is of the same value (1 kg of pure gold). However, this island is full of traps. If you take a treasure box, some other treasure boxes cannot be taken. Otherwise a bomb will destroy the island (including yourself).

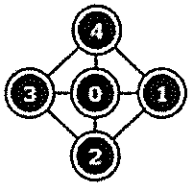
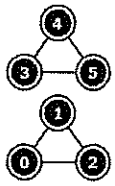
Fortunately, you somehow get a hand of the king's secret note that records the structure of the treasure boxes. The king's secret note first starts with two integers **N** and **M** in one line followed with a blank line. The king has numbered the treasure boxes from [0..**N**-1]. Each of the next **M** ($0 \leq M \leq N*(N-1)/2$) lines in the king's secret note contain two different integers **i** and **j** which means that the king has set up a trap such that if **both** treasure boxes **i** and **j** are taken away from their position, the island will self-destruct. The integer **i** and **j** also imply that there is a straight path in the island that connects the location of treasure box **i** and the location of treasure box **j**.

The **optimization version** of **TREASURE-HUNT** is this: Given a king's secret note, what is the maximum number of treasure boxes that you can take out from that island **alive**?

The **decision version** of **TREASURE-HUNT** called **TREASURE-HUNT-DEC** is this: Given a king's secret note, can we take out **K** out of **N** treasure boxes from that island **alive**?

(a). (2 points) Understanding TREASURE-HUNT

Given these five different king's secret notes, please list down the treasure boxes that you will take out from the island to solve the optimization version of TREASURE-HUNT. The first king's secret note has been solved as an example.

Secret Note 1	Secret Note 2	Secret Note 3	Secret Note 4	Secret Note 5
3 2 0 1 1 2	2 1 0 1	5 4 0 1 0 2 0 3 0 4	5 8 0 1 0 2 0 3 0 4 1 2 1 4 2 3 3 4	6 6 0 1 0 2 1 2 3 4 3 5 4 5
				
Ans: 2 boxes Take {0, 2}	Ans:__box(es) Take {_____}	Ans:__box(es) Take {_____}	Ans:__box(es) Take {_____}	Ans:__box(es) Take {_____}

(b). (2 points) Prove that TREASURE-HUNT-DEC is in NP

Hint: Let the certificate be the set of treasure boxes that you take and a target integer K.

(c). (5 points) Prove that 3-CNF-SAT $\leq_p$ TREASURE-HUNT-DEC.

For this question, you can assume that 3-CNF-SAT has been proven to be NP-Complete.

Formal definition of 3-CNF-SAT problem:
Input: A 3-CNF formula ϕ (satisfiability instance in Conjunctive Normal Form and with exactly 3 distinct literals per clause)
Output: Whether the 3-CNF ϕ is satiafiable?

Please follow the steps below:

1. Write down the general idea of your polynomial-time reduction algorithm that takes in a 3-CNF formula ϕ and convert it into an instance of TREASURE-HUNT-DEC **(1 point)**

*Hint: Consider a triangle with 3 vertices (see the king's secret note 5 in **Q2 (a)**).*

2. Apply your reduction algorithm on the following 3-CNF formula ϕ
   $\phi = (x_1 \vee x_2 \vee \neg x_4) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$
   to produce an instance of TREASURE-HUNT-DEC **(1 point)**.

3. Prove the correctness of your reduction algorithm **(2 points)**

4. Analyze the time complexity of your reduction algorithm. Is it polynomial? **(1 point)**

(d). (1 point) Conclusion

Due to the results of **Q2 (b)** & **(c)**, what can you conclude about TREASURE-HUNT-DEC?

Q3. (Special case of TREASURE-HUNT) (10 points)

(a). (3 points) Trying your luck...

Before you actually take out any treasure box (and risk blowing out yourself), you decide to explore the island first. You walk from the location of one treasure box to another treasure box in that island. To your amazement, you find out that you are always able to go from any treasure box **i** to any treasure box **j** via only one unique path where **i, j** $\in$ **[0..N-1]**. King secret's note 1, 2, and 3 in **Q2 (a)** above exhibits this property.

(a.1). (1 point) From these observations, you conclude that the connection of these treasure boxes in the island is actually a *special* graph structure called a

(circle one: Directed Acyclic Graph/Tree/Bipartite Graph).

(a.2). (1 point) You also conclude that if you run Depth First Search (DFS) algorithm on the graph that represents the connection of these treasure boxes from any starting treasure box, you will **only** encounter

(circle one: tree/back/forward/cross) edge(s).

(a.3). (1 point) You also conclude that if you run Breadth First Search (BFS) algorithm on the graph that represents the connection of these treasure boxes, you will get

(circle one: the same/different) spanning tree

compared to the DFS spanning tree. Please assume that BFS/DFS starts from treasure box 0.

(b). (7 points) You are feeling lucky!

Assuming that the property that you have observed in **Q3 (a.1)** is true, design a **Dynamic Programming** algorithm for solving this special case of TREASURE-HUNT that runs in **O(N)** time and uses at most **O(N)** space to compute.

Design a recursive formula that solves this special case of TREASURE-HUNT.

Hint: For each treasure box, you have two possible actions: Take or not take it. Start your Dynamic Programming algorithm by deciding to take or not to take treasure box 0 and then decide what you should do with the remaining treasure boxes (the subproblems).

(b.1). (2 points) List down all the recursive cases.

(you can use this space to continue your answer for (b.1))

(b.2). (1 point) List down all the base cases.

(b.3). (2 points) Show an instance of overlapping subproblems in the recursive formula that you have written down in **Q3 (b.1)** when applied to the king's secret note 3 as in **Q2 (a)**.

(b.4). (2 points) Analyze the time complexity of your Dynamic Programming solution. You need to clearly show why it runs in **O(N)** and uses at most **O(N)** table size.

Q4. (Greedy Approximation for TREASURE-HUNT) (5 points)

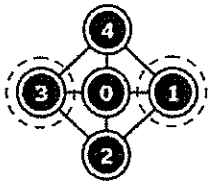
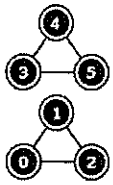
The special case that we have discussed earlier in **Q3 (a)** turns out to be non-existent, i.e. the king's secret note can still contain cases like the sample 4 or 5 shown in **Q2 (a)**.

Fearing that taking the wrong subset of treasure boxes will cause disaster, you hire Professor X from an unknown university Y. After a while, Professor X designs the following algorithm for solving the general case of TREASURE-HUNT. Professor X proudly *claims* that his algorithm is a 1-approximation algorithm, i.e. it is an optimal algorithm.

The pseudo-code of Professor X's algorithm is as follows:

1. Initially all treasure boxes are labeled as "safe";
2. Find **T**, the treasure box with the smallest number of other "safe" treasure boxes connected to it (if tie, choose the treasure box with lower box number), and then take this treasure box T; **// greedy choice**
3. Do NOT take any treasure boxes connected to **T** and mark them as "dangerous";
4. If there is at least one treasure box that is labeled as "safe", repeat step 2;

(a). (1 point) Apply Professor X's algorithm on king's secret note 1, 2, 3, 5 as in **Q2 (a)** and copied below. This time, the solution returned by Professor X's algorithm for king's secret note 4 is already shown: The algorithm first picks treasure box 1, as it has 3 treasure boxes connected to it and it has *the lowest box number*. We mark the neighboring treasure boxes {0, 2, 4} as "dangerous". We are now left with the last safe treasure box 3 and we take it.

Secret Note 1	Secret Note 2	Secret Note 3	Secret Note 4	Secret Note 5
3 2 0 1 1 2	2 1 0 1	5 4 0 1 0 2 0 3 0 4	5 8 0 1 0 2 0 3 0 4 1 2 2 3 3 4 4 1	6 6 0 1 0 2 1 2 3 4 3 5 4 5
				
Ans:__box(es) Take {_____}	Ans:__box(es) Take {_____}	Ans:__box(es) Take {_____}	Ans: 2 box(es) Take {1, 3}	Ans: __ box(es) Take {_____}

(b). (2 points) Step 2 and 3 of Professor X's pseudo code are two big steps. Please complete the design of Professor X's algorithm by using good data structure(s) so that the algorithm runs in O((N + M) * log N).

(c). (2 points) Due to your previous conclusion in **Q2 (d),** it is unlikely that Professor X's polynomial time algorithm is optimal unless P = NP. Show that Professor X's algorithm is **not** an optimal greedy algorithm by supplying one instance of the optimization version TREASURE-HUNT where the number of treasure boxes taken in the optimal solution is at least one more than the number of treasure boxes taken by Professor's X algorithm. Note that to get the points in this question, you need to clearly explain your answer!

Hint: Consider instances with N > 5 treasure boxes.

Score: (___)

Q5. (Solve small instances of TREASURE-HUNT) (5 points)

The special case that we have discussed earlier in **Q3 (a)** turns out to be non-existent, i.e. the king's secret note can still contain cases like the sample 4 or 5 shown in **Q2 (a).** Moreover, Professor X's algorithm discussed in **Q4** turns out to be sub-optimal.

Then you notice a very small footnote in the king's secret note: Hundred years ago (before computers were invented), the king had successfully asked one of his advisers (a human) to compute the answer for this TREASURE-HUNT problem **manually**. You traverse the island one more time and this time you count the number of treasure boxes. You encounter no more than $\mathbf{N} \leq 60$ treasure boxes. Now, design the best possible algorithm (or heuristic) to solve the small instances of TREASURE-HUNT problem.

Hint: Trying all $O(2^N)$ subsets to see if it is an Independent Set (doable in O(M) time) and keeping the largest subset is not possible and will receive 0 point.

-- End of Paper --