CPSC 320 Sample Final Examination
December 2013

Name: _____ Student ID: _____

Signature: _____

- You have 2.5 hours to write the 9 questions on this examination. A total of 100 marks are available.
- **Justify all of your answers, except if the question says not to.**
- No notes or electronic equipment are allowed, except for **one** 8.5×11 **sheet of paper, handwritten**.
- Keep your answers short. If you run out of space for a question, you have written too much.
- The number in square brackets to the left of the question number indicates the number of marks allocated for that question. Use these to help you determine how much time you should spend on each question.
- Use the back of the pages for your rough work.
-

Question	Marks
1	
2	
3	
4	
5	
6	
7	
8	
9	
Total	

UNIVERSITY REGULATIONS:

- Each candidate should be prepared to produce, upon request, his/her UBC card.
- No candidate shall be permitted to enter the examination room after the expiration of one half hour, or to leave during the first half hour of the examination.
- CAUTION: candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action.
    1. Having at the place of writing, or making use of, any books, papers or memoranda, electronic equipment, or other memory aid or communication devices, other than those authorised by the examiners.
    2. Speaking or communicating with other candidates.
    3. Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.
- Candidates must not destroy or mutilate any examination material; must hand in all examination papers; and must not take any examination material from the examination room without permission of the invigilator.

[10] 1. Answer each of the following questions with *true* or *false*. Give a *short* justification for each of your answers.

[5] a. $6^n \in O(5^n)$

[5] b. $1.5^n + n^2 \in O(1.5^n + n \log n)$

[18] 2. Short Answers

[3] a. Why is it important to know that a problem is NP-Complete?

[3] b. Explain why the following statement holds: "Finding the median of an unordered array is as difficult as finding the i^{th} order statistics of that array, for an arbitrary i". Hint: think of the implementations of algorithm `RandomizedQuickSelect`

[3] c. What is the main advantage of `RandomizedQuickSelect` over `Select1`?

[3] d. In our divide and conquer algorithm to find the closest pair of points in the plane, why was it sufficient, during the *merge* step, to consider points that were at most 15 positions apart in the list of points in the vertical strip around the dividing line?

[3] e. Why does the Gale-Shapley (Stable Matching) algorithm discussed in class terminate after at most n^2 iterations?

[3] f. When do we use amortized analysis?

[9] 3. A Computer Science researcher has designed a new data structure called a *sheap* that supports operations `insert`, `findMinMax` and `extractMinMax`, and decides to use amortized analysis to determine the worst-case running time of a sequence of n operations on an initially empty sheap. She defines a potential function Φ for sheaps, such that $\Phi(S) \geq 0$ for every sheap S, and such that $\Phi(S) = 0$ if the sheap S is empty. After analyzing the running time of the three operations, she has learned that

- An `insert` operation on a sheap with n elements takes time $\log n$, and increases the sheap's potential by 2.
- A `findMinMax` operation on a sheap with n elements takes time $x + \sqrt{n}$, where x is the number of elements examined by the operation. The potential of the sheap goes down by $x - 1$.
- A `extractMinMax` operation starts by performing a `findMinMax` operation. The remainder of the `extractMinMax` operation takes time $\log n$, and decreases the sheap's potential by 1.

Give as tight a bound as possible on the worst-case running time of a sequence of n operations on an initially empty sheap.

[9] 4. For each of the following recurrence relations, determine whether or not the Master Theorem discussed in class can be used. If it can be used, apply it to derive the solution of the recurrence relation using O notation. If the Master Theorem can not be used, explain why briefly.

[3] a. $T(n) = \begin{cases} 2T(\lfloor\sqrt{n}\rfloor) + n & \text{if } n \geq 2 \\ 1 & \text{if } n \leq 1 \end{cases}$

[3] b. $T(n) = \begin{cases} 9T(n/3) + 2n^2 & \text{if } n \geq 3 \\ 1 & \text{if } n \leq 2 \end{cases}$

[3] c. $T(n) = \begin{cases} 4T(\lfloor n/2 \rfloor) + n^{2+\text{odd}(n)} & \text{if } n \geq 2 \\ 1 & \text{if } n \leq 1 \end{cases}$ where $\text{odd}(n) = \begin{cases} 1 & \text{if } n \text{ is odd} \\ 0 & \text{if } n \text{ is even} \end{cases}$

[5] 5. Write a recurrence relation that describes the running time of the following algorithm as a function of n. Algorithm `DoSomethingConstant` runs in $\Theta(1)$ time, and algorithm `DoSomethingLinear` runs in $\Theta(N)$ time,

```
Algorithm MysteryP(A, N, n)
  //
  // A is an array with N element, n is a value between 0 and N.
  //
  if (n = 0) then
    DoSomethingLinear(A, N)
  else
    For i ← n-1 downto 0 do
      DoSomethingConstant(A, n, i)
      MysteryP(A, N, n-1)
      DoSomethingConstant(A, n, i)
  endif
```

You may treat N as a constant.

[9] 6. Using the guess and test method, prove that the function $T(n)$ defined by

$$T(n) = \begin{cases} 5T(n-2) + 12T(n-3) + 32 & \text{if } n \geq 4 \\ 25 & \text{if } n = 3 \\ 7 & \text{if } n = 2 \\ 1 & \text{if } n = 1 \end{cases}$$

belongs to $O(3^n)$.

[8] 7. Binomial and Fibonacci Heaps

[2] a. How do we merge a pair of binomial trees of order k?

[1] b. One of the operations supported by binomial heaps is critical, in the sense that it is needed in order to perform most of the other operations. Which one is it?

[2] c. Explain how the operation whose name you wrote in part (b) is used in the implementation of the `extractMin` operation.

[3] d. Why does the Fibonacci heap implementation of `decreaseKey` use cascading cuts?

[15] 8. The CEO of a software company wants to keep his developers happy by giving them a bonus, but does not want to spend too much money. He thus wants to select as few developers as possible (these will get a bonus, and be happy), chosen so that every other developer likes at least one of those that get a bonus (and hence will be happy for him/her).

The CEO's problem can be modeled using a graph $G = (V, E)$: each node in the graph is a developer, and an edge (u, v) means that developer u likes developer v. The CEO is looking for a minimum subset W of the set V of vertices, where for every vertex $u \notin W$, there is an edge (u, w) where $w \in W$. This is called a *minimum dominating set* for the graph G.

[9] a. Write a greedy algorithm that finds a subset W of V (it does need to be the subset with the fewest elements possible) with that property. Your mark will depend in part on the criterion you use to decide which vertex to add to W. You do not need to give pseudo-code, but you should indicate what data structure you are using to store the vertices that your algorithm has not dealt with yet.

[3] b. What is the running time of the algorithm you described in part (a)?

[3] c. There is no dynamic programming algorithm known for the minimum dominating set problem. Knowing this, what can you conclude about the greedy algorithm you gave in part (a)?

[17] 9. Anne and Simon have been arguing about where to have lunch after the CPSC 320 final examination, and decide to solve the decision by playing a sequence of Poker games. The first person to win n games gets to choose where they will have lunch. Anne is a slighly better poker player, and has a 51% chance of winning any individual Poker game.

Let $P(i, j)$ be the probability that Anne will be the first person to win n games, given that she only needs to win another i games (that is, Anne has won $n - i$ games so far), and that Simon only needs to win another j games (that is, he has won $n - j$ games so far). It can be proved that

$$P(i, j) = 0.49P(i - 1, j) + 0.51P(i, j - 1) \quad (1)$$

Anne is worried about his chances of winning, and asks you to use dynamic programming to compute the probability $P(n, n)$ that she will win n games before Simon does.

[3] a. State how big a table you need to answer Anne's question using dynamic programming, and the meaning of each entry in the table.

[3] b. Give one possible order that you can use to compute the entries in the table from part (a).

[3] c. List all of the base cases that will be needed when you are computing the entries in the table from part (a). That is, list the cases where you can not use equation (1) to compute $P(i, j)$, and the value of $P(i, j)$ for each of them.

[8] d. Write pseudo-code for an algorithm that uses dynamic programming to determine the probability that Anne will win n Poker games before Simon does.