CPSC 320 Sample Midterm 1
February 2011

Name: _____ Student ID: _____
Signature: _____

– You have 50 minutes to write the 5 questions on this examination. A total of 40 marks are available.

– **Justify all of your answers.**

– You are allowed to bring in one hand-written, double-sided 8.5 x 11in sheet of notes, and nothing else.

– 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.

Question	Marks
1	
2	
3	
4	
5	
Total	

– Use the back of the pages for your rough work.

– **Good luck!**

UNIVERSITY REGULATIONS:

– Each candidate should be prepared to produce, upon request, his/her library 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.

[13] 1. Short answers

[4] a. We frequently prove an $O(n^2)$ bound on the worst-case running time $T(n)$ of an algorithm, without proving any Ω bound on $T(n)$. However, we would never prove an $\Omega(n^2)$ bound on $T(n)$ without also proving a O bound on $T(n)$. Why not?

[4] b. Why is Dijkstra's algorithm greedy?

[5] c. Mr. Isulo, a famous alien computer scientist, has designed a greedy algorithm to solve the *Clique* problem (you don't need to know what it is) on a type of graphs called *circular arc graphs* (you don't need to know what they are either). His algorithm starts by choosing the vertex with the most neighbours.

Mr. Isulo wants to prove the following lemma: "Every circular arc graph has a maximum clique that contains the vertex with the most neighbours", but he eventually finds a counter-example to his conjecture. What does this imply for Mr. Isulo's algorithm, and why?

[5] 2. Give an example of a function $f : \mathbf{N} \to \mathbf{R}^+$ that is not in $O(n \log n)$, but is not in $\omega(n \log n)$ either. Justify your answer briefly.

[5] 3. A long string consists of the four characters A, C, G and T; they appear with frequencies 31%, 20%, 9% and 40% respectively. What code would Huffman's algorithm return for each of these characters (that is, list the sequence of bits that would be used to represent each character)? Justify your answer.

[5] 4. Recall that a simple path from a vertex s of a graph G to another vertex v of G is a path from s to G that contains each vertex at most once. A student who was interested in finding the **maximum** cost simple path from a vertex s of G to every other vertex of G decided that he could achieve this by using Dijkstra's algorithm, using a Max-Heap instead of a Min-Heap, initializing costs to $-\infty$ instead of $+\infty$, and replacing the line

```
if (Cost(v) > Cost(u) + cost(u,v)) then
    Cost(v) ← Cost(u) + cost(u,v)
```

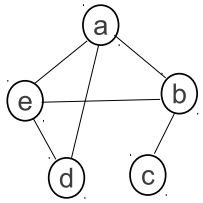
by

```
if (Cost(v) < Cost(u) + cost(u,v)) then
    Cost(v) ← Cost(u) + cost(u,v)
```

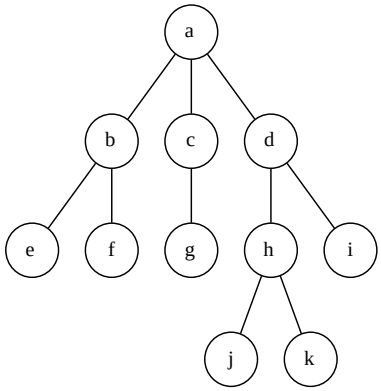
Give an example of a graph where this student's algorithm will fail. Show both the output from the modified version of Dijkstra's algorithm, and a path from s to v that is longer than the path found by the algorithm.

[12] 5. Given a graph $G = (V, E)$, and a parity function $p : V \to \{\text{"odd"}, \text{"even"}\}$ (that is, we assign a value "odd" or "even" to each vertex of G), the *parity matching* problem consists in finding a subset E' of the edges of G such that for every $v \in V$, the number of edges of E' having v as an endpoint is odd if $p(v) =$ "odd", and even if $p(v) =$ "even".

For instance, given the following graph and the parity function that assigns "odd" to vertices d and e, and "even" to vertices a, b and c, both $\{(a, b), (a, e), (b, e), (d, e)\}$ and $\{(d, e)\}$ are valid parity matchings.



[2] (a) Consider the following tree G, and the given parity function p:



even: a, b, c, g, i, j
odd: d, e, f, h, k

For each edge, determine if the edge (1) **must** belong to a parity matching of G, p, (2) **can not** belong to a parity matching of G, p, or (3) belongs to **some** parity matchings of G, p, but **not to all** of them.

i. The edge (b, f).

ii. The edge (h, j).

[8] (b) Describe a greedy algorithm that determines whether or not a **tree** $T = (V, E)$ has a parity matching for a given function p. Hint: think about part (a).

[3] (c) Analyze the worst-case running time of your algorithm from part (b).