CPSC 320 Sample Midterm 1
October 2010

[12] 1. Answer each of the questions with either *true* or *false*. You **must** justify each of your answers; an answer without a justification will be worth at most 1.5 out of 4.

[4] a. $3^{n+2} + 5 \in O(3^{n-1})$.

Solution : This is true. Here are two proofs of this fact:

- Using limits:
$$\begin{aligned} \lim_{n\to\infty} (3^{n+2} + 5)/3^{n-1} &= \lim_{n\to\infty} 3^{n+2}/3^{n-1} + \lim_{n\to\infty} 5/3^{n-1} \\ &= 27 + 0 \\ &= 27 \end{aligned}$$
Hence $3^{n+2} + 5 \in \Theta(3^{n-1})$.
- Using the definition of O: pick $c = 32$ and $n_0 = 1$. Then $3^{n+2}+5 = 27\cdot 3^{n-1}+5 \leq 27 \cdot 3^{n-1} + 5 \cdot 3^{n-1}$ (for $n \geq 1$), and so $3^{n+2} + 5 \leq 32 \cdot 3^{n-1} = c3^{n-1}$.

[4] b. Let f, g be two functions from $\mathbf{N}$ into $\mathbf{R}^+$. Assuming that $\lim_{n\to\infty} f(n)/g(n)$ exists, we can use its value to determine whether or not f is in $O(g)$.

Solution : This is true: if $\lim_{n\to\infty} f(n)/g(n) = 0$ then $f \in o(g)$, and hence $f \in O(g)$. If $\lim_{n\to\infty} f(n)/g(n)$ is a positive real number, then $f \in \Theta(g)$ and so $f \in O(g)$. Finally if $\lim_{n\to\infty} f(n)/g(n) = +\infty$ then $f \in \omega(g)$, and therefore $f \notin O(g)$.

[4] c. In class, we proved an $\Omega(n \log n)$ lower bound on the worst-case running time of any algorithm that can be used to sort a sequence of n values.

Solution : This is false: we only proved an $\Omega(n \log n)$ lower bound on the worst-case running time of any comparison sort. We did not prove that there isn't some other type of algorithm that might be able to sort a sequence of n values faster.

[8] 2. Show how to construct an input for which the prefix tree generated by Huffman's algorithm will have all of its leaves on either the bottom-most level, or the level just above it. Your construction should work for any alphabet size $m \geq 2$. Sketch a proof that it results in a prefix tree with the requested property (you do not need to write a complete proof).

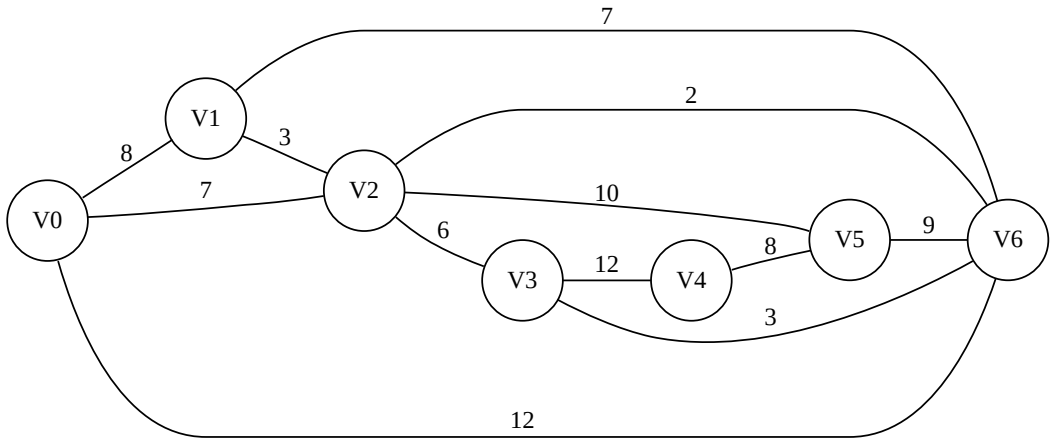
Solution : I claim that if all characters have the same frequency f, then the tree will have the desired property. The simplest proof of this fact is a proof by contradiction. Suppose that Huffman's algorithm generated a tree T with depth d, and at least one leaf at depth d' where $d' < d - 1$. Let x and y be the two characters that occupy siblings leaves at depth d, and z be the character at depth d'.

We build a tree T' from T by moving the character z from its current node to the parent of x and y, and by making x and y children of the node that used to contain z. What

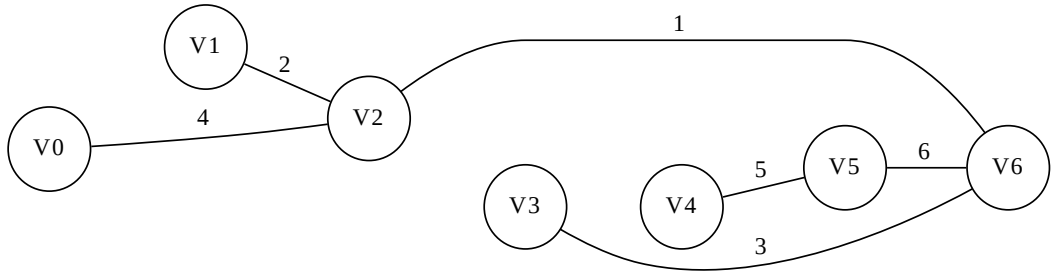
happens to the length of the code? The code grows by $d - 1 - d'$ bits for every occurrence of z, and shrinks by $d - d'$ bits for every occurrence of x and y. The total change is thus $f(d - 1 - d') - f(d - d')$, which after simplification is just $-f$.

But this means that T' gives us a code shorter than the code found by Huffman's algorithm, which is impossible. Therefore every leaf of T must have been at level d or $d - 1$.

[8] 3. Show the tree constructed by Kruskal's minimum spanning tree algorithm for the graph in the following figure. Label each edge of your tree by a number that indicates the order in which the edges were added to the tree (so the first edge added will be labeled "1", the second edge added will be labeled "2", etc).



Solution :



[12] 4. The Vancouver aquarium is undergoing major renovations. For the duration of the renovations, only one water tank will be available to the public. The aquarium's president decides to put as many different species of fish in that one tank so the public can see as many of them as possible. He asks for your help. He provides you with the following information:

- A list $F_1, \ldots, F_n$ of the n types of fish that the aquarium owns, and
- A list (F_i, F_j) of pairs of kinds of fish that can not be put in the same water tank as one would eat the other.

[9] a. Design a greedy algorithm that takes this information as input, and returns a list of species of fish that can cohabit peacefully. Your algorithm should try to maximize the number of species in the list.

Solution : Here is one possible solution:

```
Algorithm fillAquarium(F, P)
  // F is the list of species of fish.
  // P is the list of pairs of species we can not put together

  A ← ∅
  while F is not empty do
    find species f of F that occurs the fewest times in P
    remove f from F
    add f to A
    remove from F every species x such that (f, x) ∈ P
    remove all occurrences of f from P
  endwhile

  return A
```

This is all that you were required to answer on the exam. Let us now analyze the worst-case running time of the algorithm (this was **not** required).

A brute-force implementation of this algorithm will loop through P every time it needs to find the next species of fish, and will run in $O(|F||P|)$ time. A more careful implementation would work as follows:

- We count how many times each species f occurs in P. This can be done in $O(|F|+|P|)$ time by initializing a counter to 0 for each element of $|F|$, and then going through the list of pairs once, incrementing the counters for both elements of each pair. We also store all other species of fish with which a species f is incompatible.
- We then store the elements of $|F|$ in a min-heap.
- The remainder of the algorithm then performs $|F|$ `deleteMin` and `delete` operations on the heap.

This allows the algorithm to run in $O(|P| + |F| \log |F|)$ time. Note that, unfortunately, this algorithm does not always return the largest possible set of species (it is likely that no greedy algorithm can do it).

[3] b. Explain briefly why your algorithm from part (a) is greedy.

Solution : It is greedy because it always chooses the fish species whose inclusion forces the exclusion of as few other species as possible.