Basic Algorithms, Assignment 9 SOLUTIONS
Due, Thursday, Nov 15

1. Page 190, Exercise 5. For definiteness, let \( x[i], 1 \leq i \leq n \) be the locations of the \( n \) houses (in miles from the western endpoint) and let \( y[j], 1 \leq j \leq s \) be the placements of your stations. Assume the \( x[i] \) are already ordered. Design your algorithm, give a cogent argument for its correctness, and analyze its time as a function of \( n \).

Solution: The algorithm is to wait as long as possible to place the stations. The first station is placed at \( y[1] = x[1] + 4 \). Set \( L = y[1] \), the placement of the last station so far. Set \( J = 1 \), the number of stations so far. Now for \( 1 \leq i \leq n \), if \( x[i] \leq L + 4 \) you do nothing. Else you set \( J = J + 1 \) and \( y[J] = x[i] + 4 \). This is a linear time algorithm (given the sorted \( x[i] \)).

2. Suppose we are given the Minimal Spanning Tree \( T \) of a graph \( G \). Now we take an edge \( \{x, y\} \) of \( G \) which is not in \( T \) and reduce its weight \( w(x, y) \) to a new value \( w' \). Suppose the path from \( x \) to \( y \) in the Minimal Spanning Tree contains an edge whose weight is bigger than \( w \). Prove that the old Minimal Spanning Tree is no longer the Minimal Spanning Tree.

Solution: We can replace the edge whose weight is bigger than \( w \) with the edge \( \{x, y\} \) resulting in a lower weight spanning tree.

3. Let \( n = 2^t \). Consider the alphabet \( S = \{1, \ldots, n\} \) with frequencies \( f[i] = 2^{-i}, 1 \leq i \leq n - 1 \) and \( f[n] = 2^{-n+1} \). Describe how the Huffman Code Algorithm with work, the final code \( \gamma \), and \( ABL[\gamma] \), the Average Bits per Letter for the code. Let \( \gamma^* \) denote the code that sends \( i \) into the binary expansion of \( i - 1 \), where each binary expansion is given \( t \) bits. What is \( ABL[\gamma^*] \) as a function of \( n \).

Solution: For \( \gamma^* \) we have a constant length code so \( ABL[\gamma^*] = t = \lg n \). For \( \gamma \) we have the encoding \( \gamma[1] = 0, \gamma[2] = 10, \gamma[3] = 110, \ldots, \gamma[n - 1] = 1^{n-2}0 \) and \( \gamma[n] = 1^{n-1}1 \). So \( ABL[\gamma] = \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \ldots \), more precisely, including the last one,

\[
ABL[\gamma] = \sum_{i=1}^{n-1} i2^{-i} + (n-1)2^{1-n}
\]

This is roughly 2 (there is an exact formula) as

\[
\sum_{i=1}^{\infty} i2^{-i} = \sum_{i=1}^{\infty} \sum_{j \geq i} 2^{-j} = \sum_{i=1}^{\infty} 2^{-i+1} = 2
\]
4. Suppose we ran Kruskal’s algorithm on a graph $G$ with $n$ vertices and $m$ edges, no two costs equal. Suppose the $n - 1$ edges of minimal cost form a tree $T$.

(a) Argue that $T$ will be the minimal cost tree.

Solution: From Kruskal’s Algorithm we will accept all the edges of $T$. Then we have a spanning tree so no more edges are accepted.

(b) How much time will Kruskal’s Algorithm take. (Assume it stops when it finds the MST.)

Solution: We do $n$ operations $\text{UNION}[x, y]$, each takes time $O(\ln n)$ so the total time is $O(n \ln n)$.

(c) We define Dumb Kruskal. It is Kruskal without the SIZE function. For $\text{UNION}[u, v]$ we follow $u, v$ down to their roots $x, y$ as with regular Kruskal but now, if $x \neq y$, we simply reset $\pi[y] = x$. We have the same assumptions on $G$ as above. How long could dumb Kruskal take. Describe an example where it takes that long. (You can imagine that when the edge $u, v$ is given an adversary puts them in the worst possible order to slow down your algorithm.)

Solution: As $\text{UNION}[x, y]$ must take time $O(n)$ (as there are only $n$ vertices) the whole algorithm will take time $O(n^2)$. This can happen. Suppose the edges were, in order, $\{2, 1\}, \{3, 1\}, \{4, 1\}, \ldots, \{n, 1\}$. For the first edge we make $\pi[1] = 2$. The second edge we follow 1 down to root 2 and set $\pi[2] = 3$. Now for the third edge we follow 1 to 2 to root 3 and set $\pi[3] = 4$. On the $i$-th step we are taking time $\sim i$ so it is a $\Theta(n^2)$ running time.

People wish to learn to swim and at the same time to keep one foot on the ground.
– Marcel Proust