Noah Treuhaft
Algorithms In The Real World
Final Exam

Problem 1

(A) The first thing we need to send is the probablities used for the arithmetic coding. The second is the initial ordering of the symbols. the symbols.

(B) ABCAB cycles through the following symbol orderings in steady state:

```
ABC
BAC
CBA
ACB
BAC
```

This results in the following positional coding: 11222. The asymptotic bits-per-symbol for this string is its entropy, which is .4*log[2](1/.4)+.6*log[2](1/.6) = .9709505947.

(C) ABCDE cycles through the following symbol orderings in steady state:

```
AEDCB
BAEDC
CBAED
DCBAE
EDCBA
```

This produces the following positional code: 55555. The asymptotic bits-per-symbol of this string is 1*log[2](1/1) = 0.

(D) It should adapt well. What matters most is the entropy of the distances. If U(N) is the number of unique characters used in file N, then move-to-front will generate at most U(N) large distances before its behavior is no longer affected by the symbol ordering created by the previous docment. Since U(N) is (hopefully) small relative to $|N|$, the effect of those large distances should be small.

(E) Coding pairs of symbols decreases by 1/2 the number of positions that must be arithmetically coded. However, we must consider its effect on the entropy of those positions. If the entropy increases by less than a factor of two, then pair coding will yield better compression.

Positions generated by move-to-front are determined by the number of unique symbols that appear between two occurrences of the same symbol. They can be bounded above by the inverse probability, so we'd like to know the entropy of the probabilities for both schemes. Since the entropy of groups of eight characters in English is lower than that of individual characters, we would expect the entropy of character pairs to be lower as well. This means a higher variance in, and lower entropy for, symbol probabilities for character pairs. Therefore, pair coding will help get better compression.

(F) Yes, Joe hacker's code would be easy to crack. We can decode his messages into a "new" alphabet, ASCII'. Then we just have to determine the mapping between ASCII' and ASCII. We can do that by using statistical information about word frequencies in English. We select a few common words that contain most ASCII characters and are statistically likely to appear in Joe's messages. Next, we search through the ASCII' message looking for patterns that might correspond to our ASCII words. It should be very easy to identify the ASCII' space character, which will make our pattern search straightforward.

Problem 2

(A) If there were a cycle in the edges corresponding to the basic variables, then rows of the corresponding submatrix would not be independent. Here is why.

This is a generic form for a cycle.

```
x  -x
    y  -y
z      -z
```

We make x=y=z=1 through multiplication of rows with -1. When we add the first two rows, we see that the resulting two rows are equal. (Remember, x=y=z.)

```
x  y-x -y
z      -z
```

The third row is therefore a linear combination of the first and second.

A dependence between the rows of the basic-variable submatrix implies that it is not be invertible. Since we know that the submatrix is invertible, we know that it cannot contain a cycle.

(B) Only linearly-independent subsets of edges can have nonzero flows.

(C) One step of the simplex method corresponds to increasing from zero the flow on the entering variable's edge, decreasing to zero the flow on the leaving variable's edge, and adjusting other nonzero flows to satisfy the supply-demand constraints at each vertex.

Problem 3

$$
\begin{array}{lll}
\text{minimize} & \sum_{(i,j)\in E} w(i,j)m_{i,j} & \\
\text{subject to} & \sum_{l=1}^{|E|} x_{i,j,l} = 1 & v_i, v_j \in D \\
 & & i < j \\
 & x_{i,j,l} = \sum_{e_{i,a}\in E} m_{i,a}x_{a,j,(l-1)} & v_i, v_j \in V \\
 & & i < j \\
 & & l = 1, 2 \ldots |E| \\
 & x_{i,j,0} = 1 & v_i, v_j \in V \\
\text{where} & x_{i,j,l}, m_{i,j} \in \{0,1\} &
\end{array}
$$

The first set of constraints permits only one path between each pair of elements of D. $x_{i,j,l}$ is 1 if is there is a path from v_i to v_j with length l, and 0 otherwise. $m_{i,j}$ is 1 if edge (i, j) is in the minimum Steiner tree, and 0 otherwise.

Problem 4

An example of worst-case behavior for the incremental method is a distribution consisting of some points on the radius of a semicircle together with one point at the semicircle's center. (The Delaunay triangulation of this point-set looks like half of a sliced pie.) When these triangles are circumscribed, the intersection of the resulting circles is nonempty. A single point added to that intersection will lie in all circles, requiring modification of all triangles in the triangulation.

Problem 5

When $k = 0$, the depth of the Barnes-Hut tree is 0. Each time we increment k, we add one level to the tree, so k is the depth of the tree. We will place $n - 1 = 1 + 2 + 4 + 8 + \ldots + 2^k = \sum_{i=0}^{k} 2^i = 2^{k+1} - 1$ points; $k = \lg_2 n - 1$ is the depth of the tree.

Problem 6

(A) Her query vector in 2-space is [.3043478261, .2000000000]. The cosine measures of the five documents are 1.657142857 -.1571428571 -.4571428572 1.219047619 .009523809553 so she gets documents one and four.

(B) You would do this by computing A_2, adding a new row that doesn't change the rank but minimizes the distance (based on $\sum\sum(a_{i,j} - b_{i,j})^2$) from [0 1 0 0 1], and then solving for the new row of U_2 based on this.

Problem 7

(A) We should divide based on the variable that minimizes the weighted sum of the two conditional entropies based on the variable.

Specifically, let S be the set of objects to be divided; let $S_{v=x}$ be the subset of S such that variable v has the value x; and let $p(v)$ be the probablity that variable v is 1 in S. We should divide the data on the variabel v' such that $p(v')H(S_{v'=0}) + (1 - p(v'))H(S_{v'=1})$.

(B) The number of bits-per-variable needed to encode S after clustering will be $1/m + H(S_{v'=0}) + H(S_{v'=1})$, so we should not dividing if this is greater than $H(S)$.