Problem Set 7—Due Friday, May 26, 2000

Problem 1. (a) Exercise 17.3-2 (p. 344).

(b) Show that if the characters are already sorted by frequency, Huffman's algorithm can be implemented in linear time.

Problem 2. We have n programs to be stored on magnetic tape. The length of program i is ℓ_i bytes, and the frequency of access of program i is f_i. We want to arrange the programs on the magnetic tape so as to minimize the expected read time. Assume that the time to read program i is proportional to the total length of the programs up to and including i, starting on the left-hand end of the tape.

Specify a greedy algorithm for the FREQUENCY-SENSITIVE TAPE-SCHEDULING problem, analyze its running time, and prove it correct.

Problem 3. Construct the string matching automaton for the pattern $P =$ `aabab`.

Problem 4. (a) Draw out the table created by running the LOCAL ALIGNMENT algorithm given in class, run on the following strings:

```
GGACACCTTA
ATACACGTTG
```

Here is the recurrence:

$$A[i,j] = \begin{cases} \max\{ & A[i,j-1]-2, \\ & A[i-1,j-1]+p(i,j), \\ & A[i-1,j]-2, \\ & 0\}, \text{ if } i>0 \text{ and } j>0 \\ 0 & \text{ if } i=0 \text{ or } j=0. \end{cases}$$

Circle the cells that give the substrings that yield the optimal local alignment, and give the maximal local alignment.

(b) Define an *end space* to be a space that appears before the first or after the last character in a string alignment. Consider this alignment:

```
CAGCA-CTTGGATTCTCGG
---CAGCTTGG--------
```

All spaces in the second alignment are end spaces, while the space in the first alignment is not an end space. By our original scoring method, (+1 for a match, −1 for a mismatch, and −2 for a space) this is not the optimal alignment. This alignment is optimal:

```
CAGCACTTGGATTCTCGG
CAGC-----G-T----GG
```

In spite of the higher score, this second alignment is not as interesting if we're looking for regions that are similar. If we want instead the regions of a longer sequence which resemble the shorter sequence (in its entirety), we would prefer the first alignment.

Modify our dynamic programming algorithm for approximate string matching (finding an optimal alignment) to ignore end spaces. (Note: this is different from local alignment, which ignores leading and trailing mismatches also!)

Give the recurrence and analyze the running time as usual.