Spring, 2025 CIS 5110

Introduction to the Theory of Computation
Jean Gallier

Homework 4

March 21, 2025; Due April 10, 2025

Problem B1 (100 pts). (1) Implement the Viterbi algorithm, as described in Section 4.2 of the notes. **Do not implement any other version of the Viterbi algorithm found on the web or anywhere else**.

The input should consist of the matrices A (an $n \times n$ matrix), B (an $n \times m$ matrix), the vector π (of dimension n), and a sequence $(\omega_1, \ldots, \omega_T)$ of length T consisting of the indices associated with the observation sequence $(O_1, \ldots, O_T)$ (given by the bijection $\omega \colon \mathbb{O} \to \{1, \ldots, m\}$).

The output should be

(a) The sequence $(i_1, \ldots, i_T)$ of indices associated with the state sequence $(q_1, \ldots, q_T)$ (given by the bijection $\sigma \colon Q \to \{1, \ldots, m\}$) that yields the highest probability of producing the observation sequence $(O_1, \ldots, O_T)$,

(b) The highest probability $maxscore = \max_{1 \leq j \leq n} score(j, T)$ found at time T.

In Example 4.1 of the notes, we have $Q = \{\text{Cold}, \text{Hot}\}$, the bijection σ is given by $\sigma(\text{Cold}) = 1$ and $\sigma(\text{Hot}) = 2$, the output alphabet is $\mathbb{O} = \{\text{N}, \text{D}\}$, and the bijection ω is given by $\omega(\text{N}) = 1$, and $\omega(\text{D}) = 2$.

The output sequence NNND corresponds to the sequence $(1, 1, 1, 2)$, and the state sequence $(\text{Hot}, \text{Cold}, \text{Cold}, \text{Hot})$ corresponds to $(2, 1, 1, 2)$.

The matrices A, B and the vector π are given in the notes.

Test your program on the HMM of Example 4.1 of the notes for the following observation sequences:

1. NNND (**2 points**)
2. NNNDN (**2 points**)
3. NNNDNN (**2 points**)

4. NNNDNDDN (**4 points**)

   In all four cases, print the most likely sequence of states.

5. The sequence of length 1200 consisting of the following four blocks: (**5 points**)

$$\underbrace{\mathrm{N}\cdots\mathrm{N}}_{300}\underbrace{\mathrm{D}\cdots\mathrm{D}}_{300}\underbrace{\mathrm{N}\cdots\mathrm{N}}_{300}\underbrace{\mathrm{D}\cdots\mathrm{D}}_{300}$$

   Print states q_1-q_5, q_{300}-q_{304}, q_{600}-q_{604}, q_{900}-q_{904}, and q_{1196}-q_{1200}.

6. The sequence of length 2000 consisting of the following four blocks: (**5 points**)

$$\underbrace{\mathrm{N}\cdots\mathrm{N}}_{500}\underbrace{\mathrm{D}\cdots\mathrm{D}}_{500}\underbrace{\mathrm{N}\cdots\mathrm{N}}_{500}\underbrace{\mathrm{D}\cdots\mathrm{D}}_{500}$$

   Print states q_1-q_5, q_{500}-q_{504}, q_{1000}-q_{1004}, q_{1500}-q_{1504}, and q_{1996}-q_{2000}.

7. The sequence of length 2004 consisting of the following four blocks, followed by NNND: (**5 points**)

$$\underbrace{\mathrm{N}\cdots\mathrm{N}}_{500}\underbrace{\mathrm{D}\cdots\mathrm{D}}_{500}\underbrace{\mathrm{N}\cdots\mathrm{N}}_{500}\underbrace{\mathrm{D}\cdots\mathrm{D}}_{500}\mathrm{NNND}$$

   Print states q_1-q_5, q_{500}-q_{504}, q_{1000}-q_{1004}, q_{1500}-q_{1504}, and q_{1999}-q_{2004}.

For the output sequences in (6) and (7) you will find $maxscore = 0$, which means that the numbers are smaller than machine precision; you run into *underflow*.

(2) To overcome underflow, modify your program by using logarithms as suggested in Section 4.2 of the notes.

Run your new version of Viterbi on the sequences (5), (6), (7) of part (1). (**10 points**)

Hint. In (6), you should expect that $maxscore = -1.2275e{+}03$, and in (7), that $maxscore = -1.2318e + 03$.

(3) Consider the example of an HMM given online as Example-Viterbi-DNA. The set of states is $Q = \{\mathrm{L}, \mathrm{H}\}$, and the set of outputs is $\mathbb{O} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$. Assume we use the bijections $\sigma\colon Q \to \{1, 2\}$ and $\omega\colon \mathbb{O} \to \{1, 2, 3, 4\}$ given by $\sigma(\mathrm{H}) = 1$, $\sigma(\mathrm{L}) = 2$, and by $\omega(\mathrm{A}) = 1$, $\omega(\mathrm{C}) = 2$, $\omega(\mathrm{G}) = 3$, and $\omega(\mathrm{T}) = 4$. Then the probability matrices are

$$A = \begin{pmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \end{pmatrix}, \quad B = \begin{pmatrix} 0.2 & 0.3 & 0.3 & 0.2 \\ 0.3 & 0.2 & 0.2 & 0.3 \end{pmatrix}, \quad \pi = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}.$$

Verify that the sequence GGCACTGAA corresponds to the state sequence HHHLLLLLL, and that the corresponding probability is $4.2515e - 08$. (**5 points**)

Which state sequence corresponds to the DNA sequence GAGATATACATAGAATTACG, and what is the corresponding highest probability? (**5 points**)

Run both versions of your Viterbi and compare the highest probabilities. By taking the exponential of the value given by the second version you should get the probability given by the first version (not using logs). (**5 points**)

Problem B2 (60 pts). (1) Prove that the intersection, $L_1 \cap L_2$, of two regular languages, L_1 and L_2, is regular, **using the Myhill-Nerode characterization** of regular languages.

(2) Let $h\colon \Sigma^* \to \Delta^*$ be a homomorphism, as defined on page 21 of the notes. For any regular language, $L' \subseteq \Delta^*$, prove that

$$h^{-1}(L') = \{w \in \Sigma^* \mid h(w) \in L'\}$$

is regular, **using the Myhill-Nerode characterization** of regular languages.

Proceed as follows: Let $\simeq'$ be a right-invariant equivalence relation on Δ^* of finite index n, such that L' is the union of some of the equivalence classes of $\simeq'$. Let $\simeq$ be the relation on Σ^* defined by

$$u \simeq v \quad \text{iff} \quad h(u) \simeq' h(v).$$

Prove that $\simeq$ is a right-invariant equivalence relation of finite index m, with $m \leq n$, and that $h^{-1}(L')$ is the union of equivalence classes of $\simeq$.

To prove that that the index of $\simeq$ is at most the index of $\simeq'$, use h to define a function $\widehat{h}\colon (\Sigma^*/\simeq) \to (\Delta^*/\simeq')$ from the partition associated with $\simeq$ to the partition associated with $\simeq'$, and prove that $\widehat{h}$ is injective.

Prove that the number of states of any minimal DFA for $h^{-1}(L')$ is at most the number of states of any minimal DFA for L'. Can it be strictly smaller? If so, give an explicit example.

Problem B3 (40 pts). (1) Prove that the function, $f\colon \Sigma^* \to \Sigma^*$, given by

$$f(w) = w^R$$

is RAM computable by writing a RAM program for it. ($\Sigma = \{a_1, \ldots, a_N\}$).

(2) Prove that the function, $f\colon \Sigma^* \to \Sigma^*$, given by

$$f(w) = www$$

is RAM computable by writing a RAM program for it. ($\Sigma = \{a_1, \ldots, a_N\}$).

For simplicity, you may assume that $N = 2$.

You *must* run your interpreter from B4 on these two RAM programs for a few inputs. Show the two RAM programs as specified in the syntax of your interpreter in B4.

Problem B4 (80 pts). Write a computer program implementing a RAM program interpreter. You may want to assume that the instructions have five fields

N	X	`opcode`	j	Y
N	X	`opcode`	j	$N1$

with $j \in \{1, \ldots, k\}$, where k is the number of symbols in Σ, and that the opcodes are

`add` `tail` `clr` `assign` `gotoa` `gotob` `jmpa` `jmpb` `continue`

where `gota` corresponds to jump above, `gotob` to jump below, `jpma` corresponds to jump above if condition is satisfied, and `jmpb` to to jump below if condition is satisfied. Depending on the opcode, some of the fields may be irrelevant (set them to 0).

The number of input registers is n (so your memory must have at least n registers), and the total number of registers is p. The number k, n p are input to your interpreter, as well as the program to be executed (a sequence of instructions). Assume that line numbers are integers. Also, to simplify matters, you may assume that you only consider alphabets of size at most 10, so that $a_1, \ldots, a_k$ ($k \leq 10$) are represented by the digits $0, 1, \ldots, 9$.

Your program should output.

1. The input RAM program P
2. The input strings $w_1, \ldots, w_n$ to the RAM program P.
3. The value of the function being computed.
4. The sequence consisting of the memory contents and the current program counter as your interpreter executes the RAM program.

Test your interpreter on several RAM programs (and input strings), including the programs of B3.

To give you an idea of an implementation of this interpreter in `Matlab` here is the beginning of my program.

```
%
% RAM interpreter
%
% opcodes are coded numerically as follows:
%
% add = 1; tail = 2; clr = 3; assign = 4; gotoa = 5; gotob = 6; jmpa = 7;
% jpmb = 8; continue = 9
%
% Instructions have 5 fields
%  N      X    opcode    j    Y
%  N      X    opcode    j    N1
% where j corresponds to symbol a_j
% There are n input registers, a total number of p registers, and the
% alphabet size is k; symbols are coded as 1, 2, ..., k
% line numbers are nonnegative; unused line numbers are negative
% The registers are numbers 1, 2, ..., p
```

```
% The input RAM progrm is in RAMprog
% The program counter is pc
% input is a list indata  containing the n input strings
%
%
function [res, pc, regs, counter] = RAMinterp(RAMprog, n, p, k, indata)
lenprog = size(RAMprog, 1);
%
% insert your code here
%
```

To run this program I used the following input file.

```
%
%  Running RAM interpreter
%

% concatenation of two strings

indata1{1} = [1     2     1     2]
indata1{2} = [2     1     2     1     1     2     2]

[res, pc, regs] = RAMinterp(RAMconcat,2,4,2,indata1)

% string reversal

indata2{1} = [1     2     2     2     2     1     1     2     1     2]
indata3{1} = [2     2     2     2     2     1     1     1     1]

[res, pc, regs] = RAMinterp(RAMrev,1,4,2,indata2)

% f(w) = www

[res, pc, regs, counter] = RAMinterp(RAMtriple,1,5,2,indata3)
```

Here is the program to concatenate two strings.

```
%
%  a RAM program to concatenate two strings
%

RAMconcat =
[-1    3     4    0   1;
```

```
-1    4     4    0   2;
0     4     8    1   1;
-1    4     8    2   2;
-1    0     6    0   3;
1     0     1    1   3;
-1    0     2    0   4;
-1    0     5    0   0;
2     0     1    2   3;
-1    0     2    0   4;
-1    0     5    0   0;
3     1     4    0   3;
-1    0     9    0   0]
```

TOTAL: 280 points