Midterm 2 for CS 170

PRINT your name: ,

(last) (first)

SIGN your name:

WRITE your section number (e.g. 101):

WRITE your sid:

One page of notes is permitted. No electronic devices, e.g. cell phones and calculators, are permitted. Do all your work on the pages of this examination. If you need more space, you may use the reverse side of the page, but try to use the reverse of the same page where the problem is stated.

You have 80 minutes. The questions are of varying difficulty, so avoid spending too long on any one question.

In all algorithm design problems, you may use high-level pseudocode.

DO NOT TURN THE PAGE UNTIL YOU ARE TOLD TO DO SO.

Problem	Score/Points
1	/10
2	/10
3	/10
4	/10
5	/10
Total	/50

1. Compression

(a) Consider the Lempel-Ziv compression scheme where the dictionary is limited to 8 words and 3 bits are used to encode an index into the dictionary. Recall that the dictionary begins with the empty string as the sole entry with index 0. Give the Lempel-Ziv encoding for the string below and list the words in the dictionary in the order in which they were added to it.

0|1|00|01|010|10|11|001|100|011|1

< ϵ, 0, 1, 00, 01, 010, 10, 11 >

```
000 0 000 1 001 0 001 1 100 0 010 0 010 1 011 1 110 0 100 1 000 1
```

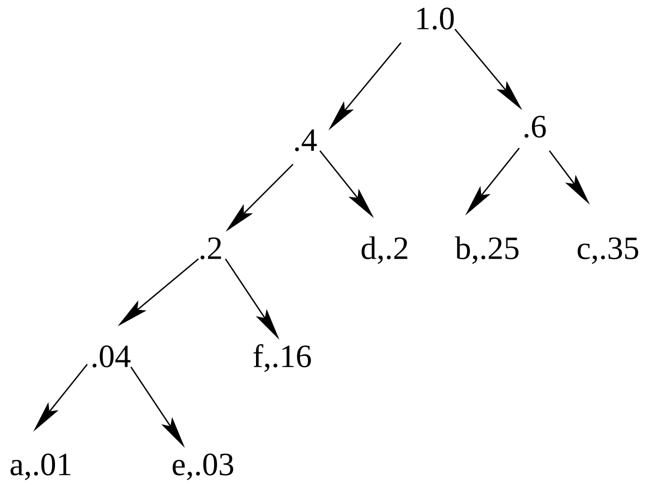
(b) Uncompress the Lempel-Ziv string below under the assumptions that each index is coded with 3 bits and the dictionary holds a maximum of 8 words. Also list, in order, the words in the dictionary.

0001|0000|0100|0110|0011|0111|0101|1001|1011|0011

< ϵ, 1, 0, 00, 000, 11, 001, 01 >

```
1 0 00 000 11 001 01 0001 111 11
```

(c) Consider the alphabet { a, b, c, d, e, f } and assume the characters occur within a file with frequencies .01, .25, .35, .20, .03, .16., respectively. Give the optimal Huffman encoding tree for this situation.



Put a 0 and a 1 label on the pair of edges out of each vertex as you please.

2. Number Theory and RSA

(a) What is $3^{28} \pmod{10}$?

$3^2 = 9 \pmod{10}$, $3^4 = 1 \pmod{10}$, $3^8 = 1 \pmod{10}$, $3^{16} = 1 \pmod{10}$

$3^{28} = 3^{16} \cdot 3^8 \cdot 3^4 = 1 \cdot 1 \cdot 1 = 1 \pmod{10}$

(b) What is the inverse of 32 (mod 113)?

From extended-Euclid's algorithm $(x', y') \to (x \bmod y, x)$ and $(a, b) \to (b', a' - b'\lfloor x/y \rfloor)$, where $ax + by = gcd(x, y)$.

x	113	32	17	15	2	1
y	32	17	15	2	1	0
a	17	-9	8	-1	1	1
b	-60	17	-9	8	-1	1

So $-60 \cdot 32 + 17 \cdot 113 = gcd(32, 113) = 1$, so $32^{-1} = -60 = 53$ (mod 113).

(c) Given a number $n = pq$ that is the product of two prime numbers p and q, how many numbers in $Z_n = \{0, 1, \ldots n-1\}$ do not have multiplicative inverses modulo n?

$p + q - 1$: ip for $i \in [1, q-1]$, iq for $i \in [1, p-1]$, and 0.

(unless $p = q$ in which case there are p such numbers. No points deducted if you missed this.)

(d) Give a decryption key (private key) for the public RSA key $(77, 7)$.

The private key $d = e^{-1}(\bmod (p-1)(q-1) = 7^{-1}(\bmod 60)$. As for problem 2.b we use Extended Euclid:

x	60	7	4	3	1
y	7	4	3	1	0
a	-5	3	-2	1	1
b	43	-5	3	-2	1

So $43 \cdot 7 - 5 \cdot 60 = gcd(7, 60) = 1$, so $7^{-1} = 43$ (mod 60).

3. Automaton Recognition

Consider a directed graph $G = (V, E)$ where each edge is labeled with a character from an alphabet Σ, and we designate a special vertex s as the start vertex, and another f as the final vertex. We say that G *accepts* a string $A = a_1 a_2 \ldots a_n$ if there is a path from s to f of n edges whose labels spell the sequence A. Design an $O((|V| + |E|)n)$ dynamic programming algorithm to determine whether or not A is accepted by G.

(a) Formally define the set of sub-problems you will solve.

For $v \in V$ and $i \in [0, n]$, let:

$A[v, i] = \exists$ path from s to v spelling $a_1 a_2 \ldots a_i$.

(b) Give your recurrence for the solution of a given sub-problem in terms of other sub-problems.

$$A[v,i] = \begin{cases} OR_{w \to v} \ \ (label(v \to w) = a_i \text{ and } A[w, i-1]) & \text{if } i > 0 \\ v = s & \text{if } i = 0 \end{cases}$$

(c) Give a non-recursive pseudo-code specification of the algorithm.

```
$A[s,0]$ ←**true**
**for** $v \in V - \{s\}$ **do**
    $A[v,0]$ ←**false**
**for** $i \leftarrow 1 \ldots n$ **do**
    **for** $v \in V - s$ **do**
      {  $A[v,i] \leftarrow$ **false**
         **for** $w \to v \in E$ **do**
             **if** $label(w \to v) = a_i$ **and** $A[w, i-1]$ **then**
                 $A[v,i] \leftarrow$ **true**
      }
**if** $A[t,n]$ **then**
    **print** "Accept"
**else**
    **print** "Reject"
```

4. Palindrome Parsing

A *palindrome* is a word $w_1w_2\dots w_k$ whose reverse $w_kw_{k-1}\dots w_1$ is the same string, e.g. "abbabba". Consider a string $A = a_1a_2\dots a_n$. A partitioning of the string is a *palindrome partitioning* if every substring of the partition is a palindrome. For example, "aba|b|bbabb|a|b|aba" is a palindrome partitioning of "ababbbabbababa". Design a dynamic programming algorithm to determine the coarsest (i.e. fewest cuts) palindrome partition of A.

(a) Formally define the set of sub-problems you will solve.

For all $1 \leq i \leq j \leq n$ we solve *two* subproblems:

$P[i,j] = a_ia_{i+1}\dots a_j$ is a palindrome (true/false), and

$C[i,j] =$ no. of cuts in the best palindrome partition of $a_ia_{i+1}\dots a_j$.

(b) Give your recurrence for the solution of a given sub-problem in terms of other sub-problems.

$$P[i,j] = \begin{cases} a_i = a_j \text{ and } P[i+1,j-1] & \text{if } i < j-1 \\ a_i = a_j & \text{if } i = j-1 \\ \text{true} & \text{if } i = j \end{cases}$$

$$\begin{aligned} C[i,j] = \min\{ \quad & \min_{k\in[i,j-1]}\{C[i,k] + C[k+1,j]\}, \\ & \text{if } P[i,j] \text{ then } 0 \text{ else } \infty\} \end{aligned}$$

(c) Give a non-recursive pseudo-code specification of the algorithm and state its complexity in terms of n.

```
for $i \leftarrow 1 \dots n$ do
    $(P[i,i], C[i,i]) \leftarrow (\textbf{true}, 1)$
for $l \leftarrow 1 \dots n-1$ do
    for $i \leftarrow 1 \dots n-l$ do
        {  $j \leftarrow i + l$
           if $l > 1$ then
               $P[i,j] \leftarrow (a_i = a_j)$ and $P[i+1,j-1]$
           else
               $P[i,j] \leftarrow (a_i = a_j)$
           $C[i,j] \leftarrow \infty$
           for $k \leftarrow i \dots j-1$ do
               $C[i,j] \leftarrow \min\{C[i,j], C[i,k] + C[k+1,j]\}$
           if $P[i,j]$ then
               $C[i,j] \leftarrow 0$
        }
Best palindrome partition has $C[1,n]$ cuts
```

The algorithm is $O(n^3)$. (This is the complexity you were expected to acheive.)

4. Palindrome Parsing

A *palindrome* is a word $w_1 w_2 \ldots w_k$ whose reverse $w_k w_{k-1} \ldots w_1$ is the same string, e.g. "abbabba". Consider a string $A = a_1 a_2 \ldots a_n$. A partitioning of the string is a *palindrome partitioning* if every substring of the partition is a palindrome. For example, "aba|b|bbabb|a|b|aba" is a palindrome partitioning of "ababbbabbababa". Design a dynamic programming algorithm to determine the coarsest (i.e. fewest cuts) palindrome partition of A.

(a) Formally define the set of sub-problems you will solve.

For all $1 \leq i \leq j \leq n$ we solve the subproblems:

$P[i,j] = a_i a_{i+1} \ldots a_j$ is a palindrome (true/false), and

and for all $0 \leq i \leq n$ we solve the subproblems:

$C[i] =$ no. of cuts in the best palindrome partition of $a_1 a_2 \ldots a_i$.

(b) Give your recurrence for the solution of a given sub-problem in terms of other sub-problems.

$$P[i,j] = \begin{cases} a_i = a_j \text{ and } P[i+1, j-1] & \text{if } i < j-1 \\ a_i = a_j & \text{if } i = j-1 \\ \text{true} & \text{if } i = j \end{cases}$$

$$C[i] = \begin{cases} \min_{k \in [0,i-1]} \{C[k] + 1 : P[k+1, i]\} & \text{if } i > 0 \\ 0 & \text{if } i = 0 \end{cases}$$

(c) Give a non-recursive pseudo-code specification of the algorithm and state its complexity in terms of n.

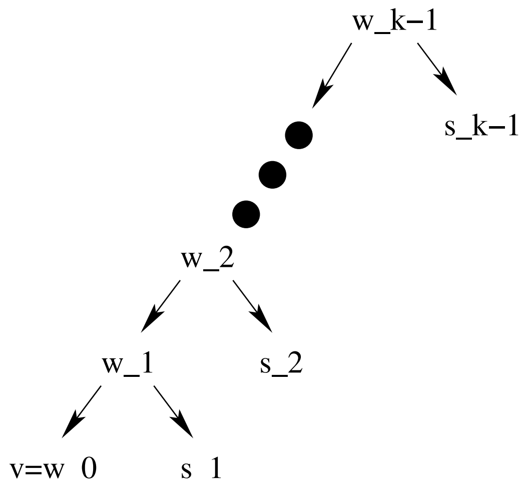
```
$P[1,1] \leftarrow$ true
for $j \leftarrow 2 \ldots n$ do
  {  $P[j,j] \leftarrow$ true
     $P[j-1,j] \leftarrow (a_{j-1} = a_j)$
     for $i \leftarrow j-2 \ldots 1$ do
          $P[i,j] \leftarrow (a_i = a_j)$ and $P[i+1, j-1]$
  }
$C[0] \leftarrow 0$
for $i \leftarrow 1 \ldots n$ do
  {  $C[i] \leftarrow C[i-1] + 1$
     for $k \leftarrow 1 \ldots i-2$ do
          $C[i] \leftarrow \min\{C[i], C[k] + P[k+1, i]\}$
  }
Best palindrome partition has $C[n]$ cuts
```

The algorithm is $O(n^2)$. (Congratulations if you got this one.)

5. Entropy and Huffman Codes [Hard]

Suppose that for alphabet Σ the probability of seeing character a is p_a. Recall the formula for the entropy of the alphabet $E(\Sigma) = \sum_{a\in\Sigma} p_a \log_2 \frac{1}{p_a}$. We asserted in class that the length of a Huffman encoded file is $O(nE(\Sigma))$. The essential fact needed to show this is that for a character a, its depth in an optimal Huffman tree is $O(\log \frac{1}{p_a})$. Prove this with a careful argument (*Hint:* Consider a node with this frequency and think about what must be true of the frequencies of all the children immediately off the path from this node to the root.)

Suppose that the leaf node v for character a is placed at depth k in an optimal Huffman tree. Let $w_0 = v, w_1, w_2, \dots w_{k-1}$ be the sequence of nodes encountered from v going back to the root of the tree, and let s_i be the child of w_i not on this path. Let $p(v)$ be the probability of the *tree* rooted at v. The figure illustrates.



Consider s_j for any $j \geq 2$. Because s_{j-1} and w_{j-2} are joined before s_j is joined to the subtree rooted at w_{j-1} it must be that $p(s_j) \geq p(s_{j-1})$ and $p(s_j) \geq p(w_{j-2})$. Therefore, $p(s_j) \geq (p(s_{j-1}) + p(w_{j-2}))/2 = p(w_{j-1})/2$ and $p(w_j) = p(w_{j-1}) + p(s_j) \geq 1.5p(w_{j-1})$. By induction it then follows that $p(w_{k-1}) \geq 1.5^{k-2}p(w_1)$. But $p(w_1) \geq p_a$ and $p(w_{k-1}) = 1$, so it must be that $p_a 1.5^{k-2} \leq 1$ which implies that $k \leq \log_1 .51/p_a + 2 = O(\log 1/p_a)$.