Midterm I. (22C:231, Spring 2011)
Open Book and Notes, total points = 100

1. **Greedy Algorithms** (25)

(a) (10) What is an optimal Huffman code for the following set of letters whose frequencies are based on the first 8 Fibonacci numbers?

- \( a : 1 \)
- \( b : 1 \)
- \( c : 2 \)
- \( d : 3 \)
- \( e : 5 \)
- \( f : 8 \)
- \( g : 13 \)
- \( h : 21 \)

Can you generalize your answer to find the optimal code when the frequencies are the first \( n \) Fibonacci numbers?

**Answer Key:** The Huffman code for these characters are (if we switch 0 and 1, we get another code; the codes of \( a \) and \( b \) can also be exchanged):

- \( C(h) = 1 \)
- \( C(f) = 01 \)
- \( C(e) = 0001 \)
- \( C(d) = 00001 \)
- \( C(c) = 000001 \)
- \( C(b) = 0000001 \)
- \( C(a) = 0000000 \).

The generalized case is that the Huffman codes are 1, 01, 001, ..., 0\( k-1 \), 0\( k \) for the characters from high frequency to low frequency, assuming we have \( k+1 \) characters in total. The code tree is a linear structure such that each level has exactly one leaf, except the root which has no leaves and the bottom level which has two leaves. The characters with lower frequencies stay at lower level.

(b) (15) Suppose a data file contains a sequence of 8-bit characters such that all 256 characters are about as common: the maximum character frequency is less than twice the minimum character frequency. Prove that Huffman coding in this case also produces 8-bit code for each character.

**Answer Key:** The Huffman algorithm starts with a set of trees of singleton, and then merge two trees with lowest frequencies into a larger tree (called composed trees). We show that if the maximum character frequency is less than twice the minimum character frequency, and we have \( n = 2^k \) characters, then every character has the Huffman code of length \( k \). We prove this by induction on \( k \).

When \( k = 1 \), we have two characters and they both have a Huffman code of length one. The statement is true.

For the inductive case, please take note that singletons will be chosen first before any composed tree in the merge process, because singletons will always have lower frequencies than that of composed trees. That is, if the minimum character frequency is \( m_1 \) and the maximum character frequency is \( m_2 \), than the frequency of any composed tree is at least \( 2m_1 \) and \( 2m_1 > m_2 \) by the assumption. After \( n/2 \) merges, all the \( n \) singletons have been chosen and we have \( n/2 = 2^{k-1} \) composed trees of two characters. If we replace those \( n/2 \) composed trees by \( n/2 = 2^{k-1} \) new characters, they are still satisfy the property that the maximum character frequency is less than twice the minimum character frequency. By the induction hypothesis, these new characters will have Huffman code of length \( k-1 \). Now replace the new characters by the corresponding composite trees of two leaves in the code tree for \( n/2 \) new characters, we have a Huffman code tree for each old character and the code length will be \( k \).

When \( n = 256 = 2^8 \), we have the solution to the original problem.

2. **Greedy Algorithms and Dynamic Programming** (25)

Given two strings of letters, say \( X = x_1x_2\cdots x_n \) and \( Y = y_1y_2\cdots y_m \), a common subsequence of \( X \) and \( Y \) is a string which appears as subsequence in both \( X \) and \( Y \). For example,
if $X = ACCGGTA$ and $Y = CGTTAG$, then $CGTA$ is a common subsequence of $X$ and $Y$. Please design an algorithm as efficient as possible to compute the length of a longest common subsequence of $X$ and $Y$. You need to present your algorithm in pseudo-code and explain briefly each line of the code. You also need to provide a time and space complexity analysis of your algorithm (using the big-O notation).

**Answer Key:** Let $Opt(i, j)$ be the longest length of common subsequence of $x_1x_2\cdots x_i$ and $y_1y_2\cdots y_j$. Then $Opt(0, j) = Opt(i, 0) = 0$ and

$$Opt(i, j) = \begin{cases} 1 + Opt(i-1, j-1) & \text{if } x_i = y_j, \\ \max(Opt(i-1, j), Opt(i, j-1)) & \text{if } x_i \neq y_j \end{cases}$$

Let $M$ be an $(n + 1) \times (m + 1)$ array to store $Opt(i, j)$, the pseudo-code will be as follows:

1. for (int $i = 0$; $i <= n$; $i++$) $M[i, 0] = 0$;
2. for (int $j = 0$; $j <= m$; $j++$) $M[0, j] = 0$;
3. for (int $i = 1$; $i <= n$; $i++$)
   4. for (int $j = 1$; $j <= m$; $j++$)
      5. if ($X[i] == Y[j]$) $M[i, j] = 1 + M[i-1, j-1]$;
      7. else $M[i, j] = M[i-1, j]$;
8. return $M[n, m]$;

Line 1 and 2 computes $M$ according to $Opt(0, j) = Opt(i, 0) = 0$.
Line 3-4 are two nested loops for other items of $M$ according to the general case of $Opt(i, j)$.
Line 5 checks if $x_i = y_j$, and let $Opt(i, j) = 1 + Opt(i-1, j-1)$.
Line 6-7 computes $\max(Opt(i-1, j), Opt(i, j-1))$ for $Opt(i, j)$ when $x_i \neq y_j$.
Line 8 returns the result $Opt(n, m)$.

The time and space complexity is $O(mn)$. The first index of $M$ can be taken modulo 2, thus the space complexity can be reduced to $O(m)$.

3. **Dynamic Programming** (25)
   You are given a $n \times n$ checkerboard and a checker. A square of the checkerboard is specified by $(x, y)$, where $1 \leq x, y \leq n$. In the beginning, the checker is at square $(x_0, 1)$ for some $1 \leq x_0 \leq n$. At each step, you must move from $(x, y)$ to $(x', y')$, where $1 \leq x', y' \leq n$, $|x - x'| \leq 1$ and $y' = y + 1$. For each move, you receive $p(x, y, x' - x)$ dollars, where $p$ is pre-defined for all $1 \leq x \leq n$, $1 \leq y < n$, and $|x - x'| \leq 1$. Please design an efficient algorithm which computes the maximum sum of dollars for all the moves from row $y = 1$ to row $y = n$. Your algorithm is free to pick any square in row 1 as a starting point and any square in the last row as a destination. Please present your algorithm in pseudo-code with brief explanation and analyze the running time of your algorithm.

**Answer Key:** Let $Opt(i, j)$ denote the maximum total dollars earned from square $(i, j)$ to the last row. Then $Opt(i, n) = 0$ for any square in the last row. For any square not in the last row, i.e. for any $1 \leq i \leq n$ and $1 \leq j < n$,

$$Opt(i, j) = \max \left\{ \begin{array}{l} Opt(i-1, j+1) + p(i, j, -1), \text{ if } i > 0 \\
Opt(i, j+1) + p(i, j, 0), \\
Opt(i+1, j+1) + p(i, j, 1), \text{ if } i < n \end{array} \right. \right.$$  

The final solution will be $\max_{1 \leq i \leq n} \{Opt(i, 1)\}$.
If we use an $n \times n$ array $M$ to store $Opt(i, j)$, the pseudo-code will be as follows:
1 for (int i = 1; i <= n; i++) M[i, n] = 0;
2 for (int j = n-1; j >= 1; j--)
3   for (int i = 1; i <= n; i++) {
5     M[i,j] = M[i,j+1] + p(i,j,0);
6     if (i>0 && M[i,j]<M[i-1,j+1]+p(i,j,-1)) M[i,j] = M[i-1,j+1]+p(i,j,-1);
7     if (i<n && M[i,j]<M[i+1,j+1]+p(i,j, 1)) M[i,j] = M[i+1,j+1]+p(i,j, 1);
8   }
9 int max = M[1,1];
10 for (int i = 2; i <= n; i++) if (max < M[i, 1]) max = M[i, 1];
11 return max;

Line 1 computes the special case of \( Opt(i, n) \); Line 2-8 compute the general case of \( Opt(i, j) \). Line 9-10 computes the maximum of \( Opt(i, 1) \) for \( 1 \leq i \leq n \). Line 11 returns the required result.

The time complexity is \( O(n^2) \).

4. Network Flow (25)
Let \( G = (X,Y,E) \) be a bipartite graph, where \( X = \{x_1, x_2, x_3, x_4\} \), \( Y = \{y_1, y_2, y_3, y_4\} \), and \( E = \{(x_1,y_1), (x_1,y_2), (x_2,y_1), (x_2,y_3), (x_3,y_2), (x_3,y_3), (x_4,y_4)\} \). We try to find a maximum matching of \( G \) using the max-flow algorithm by adding source \( s \) and sink \( t \) to \( G \) with proper capacity for each edge.

(a) (15) Suppose the first two augmenting paths we considered are \( s - x_1 - y_1 - t \) and \( s - x_3 - y_3 - t \). Please continue to find more augmenting paths until it does not exist.

**Answer Key:** The next two augmenting paths are \( s - x_2 - y_1 - x_1 - y_2 - t \) and \( s - x_4 - y_3 - x_3 - y_4 - t \).

(b) (10) Please list all the minimum cuts of the corresponding flow network.

**Answer Key:** There are five minimum cuts \( (A,B) \):

i. \( A = \{s\} \) and \( B = V - A \);
ii. \( A = \{s, x_4, y_3\} \) and \( B = V - A \);
iii. \( A = \{s, x_2, y_1, x_4, y_3\} \) and \( B = V - A \);
iv. \( B = \{t\} \) and \( A = V - B \);
v. \( B = \{t, x_3, y_4\} \) and \( A = V - B \).