University of Nevada, Las Vegas Computer Science 477/656 Spring 2023

Assignment 3: Due Saturday February 25, 2023, 11:59 PM

Name:____

You are permitted to work in groups, get help from others, read books, and use the internet. You will receive a message from the graduage assistant, Sandeep Maharjan, telling you how to turn in the assignment.

1. Compute the Levenshtein edit distance between `kitchen`" and "`chicken`." Show the matrix.

2. The standard recursive definition of the Fibonacci sequence:

   - $F(0) = 0$
   - $F(1) = 1$
   - $F(n) = F(n-2) + F(n-1)$ for $n \geq 2$.

   defines the standard dynamic program for the sequence. However, if you only want to compute F_n for a given n, there is a recurrence that allows faster computation:

   - $F(0) = 0$
   - $F(1) = 1$
   - $F(2) = 1$
   - $F(n) = F((n-1)/2)^2 + F((n+1)/2)^2$ if $n \geq 2$ is odd
   - $F(n) = F(n/2)(F(n/2-1) + F(n/2+1))$ if $n \geq 2$ is even

   Give an $O(n \log n)$ time algorithm for computing $F(n)$ for a given n. This algorithm does not require computing all earlier Fibonacci numbers, only some of them.

3. Consider problem of finding the maximum sum monotone increasing subsequence of

   5, 23, 13, 16, 8, 9, 0, 21, 15, 11, 19, 12, 6

   Draw a figure of the corresponding directed acyclic graph, as defined in the handout `dynamic.pdf`.

4. Write an algorithm that solves the following problem. Given a sequence of coins of positive value, find the maximum value set of coins which do not contain any three coins that are consecutive in the initial sequence. You need to describe the algorithm in English, not C++ code. (See the handout `dynamic.pdf`)

5. For this problem, see the handout `recursive.pdf`. Solve the following recurrences, giving the answers using Θ notation.

   Use the anti-derivative method:

   (a) $F(n) = F(n-1) + n^3$

   (b) $F(n) = F(n - \sqrt{n}) + \sqrt{n}$

   Use the master theorem:

   (c) $F(n) = F(n/4) + 1$

   (d) $F(n) = 2F(n/4) + \sqrt{n}$

   (e) $F(n) = 4F(n/2) + n$

   (f) $F(n) = 4F(n/2) + n^3$

6. Consider the following recursive function.

```
int f(int n)
 // input condition: n > 0
 {
  if(n <= 2) return n;
  else return f(n/2)+f(n/2+1)+n;
 }
```

   (a) Using the master theorem, compute the asymptotic complexity of the function $f(n)$, expressing the answer using Θ notation.

(b) Let $T(n)$ be the time it takes to compute $f(n)$ using the recursive function above. Using the master theorem, compute the asymptotic complexity of the function $T(n)$, expressing the answer using Θ notation.

(c) What follows is a portion of a C++ program which computes $f(n)$ using dynamic programming. Fill in the missing code in the `main()`.

```
int const n;
int f[n+1];

int main()
 {
  f[1] = 1;
  f[2] = 2;
  // FILL IN THE MISSING CODE HERE





  cout << "f(" << n << ") = " << f[n] << endl;
  return 1
 }
```

(d) What is the asymptotic time complexity of your code?