Problem set 2.

1. Finish the Knapsack problem started in Lecture by modifying the algorithm to **find** the subset with maximum value (recall that $M(n, b)$ just gives the maximum value but not the subset).

2. Denote the maximum number of steps taken by an algorithm on inputs of length n as $L(n)$. Assume that the structure of the algorithm implies that $L(n) \leq 2L(\frac{n}{2}) + cn$ if $n \geq 2$ and $L(1) = 1$. Show that then $L(n) \in O(n \log n)$. Is it true that $L(n) \in O(n^2)$?

3. Give a dynamic programming algorithm that, given an integer n as input, computes F_n, the n^{th} Fibonacci number in $O(n)$ steps. The sequence of Fibonacci numbers is defined by $F_0 = F_1 = 1$ and the recursion $F_n = F_{n-1} + F_{n-2}$ if $n \geq 2$. Please be sure to state clearly the 8 steps mentioned in class.

4. We are given $n+1$ positive integers ($n \geq 1$): $s_1, s_2, \ldots, s_n, b$, we want to decide whether the number b is the sum of some subset of the numbers s_i. (Each s_i can be used at most once in the sum.)
   Give a dynamic programming algorithm with running time $O(n \cdot b)$ for this problem. (Note: this is called the subset-sum problem). Use the following subproblems:
   Let $L(i, c)$ (for $0 \leq i \leq n$ and $0 \leq c \leq b$) denote whether it is possible to express the number c as a sum of some subset of $s_1, \ldots, s_i$.

---

5. Denote the maximum number of steps taken by an algorithm on inputs of size n as $L(n)$. Consider the following three recurrences for $L(n)$ and compare how the running time changes with the multiplicative factor. c is a constant, and assume $L(1) = 1$ and that the recurrences are for $n \geq 2$.
   (a) $L(n) \leq L(\frac{n}{2}) + cn$ (b) $L(n) \leq 4L(\frac{n}{2}) + cn$ (c) $L(n) \leq 2L(\frac{n}{2}) + cn$

6. A ladder has n steps. Unfortunately, some of these steps are so rickety that if we step on them, they fall off. Fortunately, we know which steps are rickety and we will not step on these. Assuming that we can in one move cover upto 3 steps on the ladder. Give me a dynamic programming algorithm that determines (a) if we can go from the bottom of the ladder (the floor) to the top step of the ladder. (b) the numer of ways in which we can get from the bottom of the ladder to the top step.
   (Assume that the top step is not rickety.) What is the running time of the algorithm?

7. We are given n positive integers, $a_1, a_2, \ldots, a_n$. Consider a table T with $n+1$ rows indexed from 0 to n, and $b+1$ columns indexed from 0 to b. Let $T[0, 0] = 1$ and $T[0, c] = 0$ for each $1 \leq c \leq b$. Give a procedure that fills in the remaining fields of T in running time $O(nb)$, such that the value of $T[i, c]$ indicates the number of ways in which some subset of the numbers $a_1, a_2, \ldots, a_i$ can be added to produce the number c ($1 \leq i \leq n, 1 \leq c \leq b$).

---

8. Denote the maximum number of steps taken by an algorithm on inputs of size n as $L(n)$. Consider the following three recurrences for $L(n)$ and compare how the running time changes with the multiplicative factor. c is a constant, and assume $L(1) = 1$ and that the recurrences are for $n \geq 2$.
   (a) $L(n) \leq 2L(\frac{n}{2}) + c$ (b) $L(n) \leq 2L(\frac{n}{2}) + cn$ (c) $L(n) \leq 2L(\frac{n}{2}) + cn^2$

9. Consider a chessboard of size $n \times n$. Unfortunately, some holes have been punched into some squares. We want to travel from the bottom left corner to the top right corner, such that in one move we can go from a square to an adjacent square above it or to its right. Provide a procedure using dynamic programming that computes the number of ways to reach the top right corner.

10. Denote the maximum number of steps taken by an algorithm on inputs of size n by $L(n)$. Give an upper bound on the magnitude of $L(n)$ if we know that $L(1) = 2$ and for $n > 1$,

    (a) $L(n) = L(n-1) + 3$
    (b) $L(n) = L(n-1) + 5$
    (c) $L(n) = L(n-1) + 3n$
    (d) $L(n) = 2L(n-1) + 3$
    (e) $L(n) = L(\lceil \frac{n}{2} \rceil) + 3$
    (f) $L(n) = L(\lceil \frac{n}{2} \rceil) + n^k$
    (g) $L(n) = 2L(\lceil \frac{n}{2} \rceil) + 3$
    (h) $L(n) = 4L(\lceil \frac{n}{2} \rceil) + 3$

    What changes if we substitute equality with $\leq$ or $\geq$?

11. Consider a recursive algorithm for finding the Fibonacci numbers. We again let $F(0) = F(1) = 1$ and then we define $F(n) = F(n-1) + F(n-2)$ for all $n \geq 2$. When we try to find $F(n)$ recursively this way, show that the running time is $\Omega(2^{\frac{n}{2}})$.

(Answers for Problem 5: (a) $O(n)$, (b) $O(n^2)$, (c) $O(n \log n)$).

(Answers for Problem 8: (a) $O(n)$, (b) $O(n \log n)$, (c) $O(n^2)$).

(Answers for Problem 10: (a) $O(n)$, (b) $O(n)$, (c) $O(n^2)$, (d) $O(2^n)$, (e) $O(\log n)$, (f) $O(n^k)$, (g) $O(n)$, (h) $O(n^2)$, these are true even if replaced with $\leq$, but not with $\geq$. In that case we can only give Ω estimates.)