Exam Coverage

The second exam will cover everything up to now, but the emphasis is on the material that we have covered after the first exam.

- Time complexity of algorithms (Section 3.3)
- Sequences and sums (Section 2.4)
- Proof by induction and strong induction (Sections 5.1 and 5.2)
- Inductive sets, recursive functions (Section 5.3)
- Proof by structural induction (Section 5.3)
- Counting (Sections 6.1 — 6.4)
- Recurrence relations and generating functions (Sections 8.1 and 8.4)
Strategy for Exam Preparation

- Start studying now (unless have already started)!
- Study class notes and make sure you know your definitions!
- Review homework, quizzes/in-class exercises
- Study the examples in the textbook
- Do odd numbered exercises
- Bring a scantron: 8x11 gray
Complexity
def bubble_sort(list)

    list = list.dup  # make copy of input

    for i in 0..(list.length-2) do
        for j in 0..(list.length - i - 2) do

            list[j], list[j + 1] = list[j + 1], list[j]  if list[j + 1] < list[j]

        end

    end

    return list

end
Nested For Loops

for i in 0..(list.length-2) do # number of iterations: n-1
  for j in 0..(list.length - i - 2) do # number of itns: n-1, n-2,..., 1
    list[j], list[j + 1] = list[j + 1], list[j] if list[j + 1] < list[j]
  end
end

Let n = list.length. Then the two nested loops take
(n-1 + n-2 + ... + 1) \( O(1) = (n(n-1)/2) \) \( O(1) = O(n^2) \) time.
Total Time Complexity

def bubble_sort(list) # O(1) for parameter assignment
    list = list.dup # O(n)
    for i in 0..(list.length-2) do # O(n^2)
        for j in 0..(list.length - i - 2) do
            list[j], list[j + 1] = list[j + 1], list[j] if list[j + 1] < list[j]
        end
    end
    return list # O(1)
end # O(1) + O(n) + O(n^2) + O(1) = O(n^2)
Weak and Strong Induction
Induction is an axiom which allows us to prove that certain properties are true for all positive integers (or for all nonnegative integers, or all integers $\geq$ some fixed number)
Induction Principle

Let $A(n)$ be an assertion concerning the integer $n$. If we want to show that $A(n)$ holds for all positive integer $n$, we can proceed as follows:

**Induction basis:** Show that the assertion $A(1)$ holds.

**Induction step:** Show that $[A(n) \rightarrow A(n+1)]$ holds for all positive integers $n$. 
Strong Induction Principle

Let $A(n)$ be the assertion concerning the integer $n$.

If we want to prove it for all $n \geq 1$, we can do the following:

**Induction basis:**

Show that $A(1)$ is true.

**Induction step:**

Show that $[(A(1) \land \ldots \land A(n)) \rightarrow A(n+1)]$ holds for all $n \geq 1$. 
You might wonder which form of induction you should use.

Generally, you use strong induction when assuming that the assertion $A(n)$ holds does not seem to help in proving $A(n+1)$. Strong induction can make the induction step easier to prove in such cases.

On the other hand, weak induction is sometimes more straightforward to use (so it is preferred when the induction step can be proved by assuming $A(n)$ alone).
Example (1/4)

Theorem: For all positive integers $n$, we have that $12$ divides $n^4 - n^2$.

Weak or strong induction?
Unclear. So let’s try weak induction.
Example (2/4)

Let's try to prove it by weak induction:

Base Step: Since $1^4-1^2 = 1 - 1 = 0 = 0*12$ is divisible by 12, the claim is true for n=1.

Inductive Step: Let us assume that 12 divides $n^4-n^2$.

Our goal is to show that this implies that 12 divides $(n+1)^4-(n+1)^2$.

Since $(n+1)^4-(n+1)^2 = n^4+4n^3+6n^2+4n+1 - (n^2+2n+1)$, we get

$(n+1)^4-(n+1)^2 = n^4 - n^2 + 4n^3+6n^2+2n$.

By assumption 12 divides $n^4 - n^2$.

Why does 12 divide $4n^3+6n^2+2n$? You don't know?

I don't know either. At least, it is not obvious.
Example (3/4)

Let us try to prove it by strong induction. Perhaps we can show that $A(n)$ implies $A(n+6)$.

**Base case:**

$n = 1$: $1^4 - 1^2 = 1 - 1 = 0 = 0*12$

$n = 2$: $2^4 - 2^2 = 16 - 4 = 12 = 1*12$

$n = 3$: $3^4 - 3^2 = 81 - 9 = 72 = 6*12$

$n = 4$: $4^4 - 4^2 = 256 - 16 = 240 = 20*12$

$n = 5$: $5^4 - 5^2 = 625 - 25 = 600 = 50*12$

$n = 6$: $6^4 - 6^2 = 1296 - 36 = 1260 = 105*12$

Thus, the claim is true for $n=1,...,6$. 
Let \( n \geq 6 \). Let us assume that 12 divides \( m^4 - m^2 \) for \( 1 \leq m \leq n \).
Our goal is to show that 12 divides \( (n+1)^4 - (n+1)^2 \).

For simplicity, let \( m = n+1-6 = n-5 \). By assumption, 12 divides \( m^4 - m^2 \).

\[
(n + 1)^4 - (n + 1)^2 = (m + 6)^4 - (m + 6)^2
= m^4 + 24m^3 + 216m^2 + 864m + 1296 - (m^2 + 12m + 36)
= (m^4 - m^2) + 24m^3 + 216m^2 + 852m + 1260
= 12(a + 2m^3 + 18m^2 + 71m + 105)
\]

for some integer \( a \). Thus, the claim follows by strong induction.
Example 2

**Theorem:** For all positive integers $n$, we have

$$1+3+5+\ldots+(2n-1) = n^2$$

Proof. We prove this by induction on $n$. Let $A(n)$ be the assertion of the theorem.

Induction basis: Since $1 = 1^2$, it follows that $A(1)$ holds.

Induction step: Suppose that $A(n)$ holds. Then

$$1+3+5+\ldots+(2n-1)+(2n+1) = n^2+2n+1 = (n+1)^2$$

holds. In other words, $A(n)$ implies $A(n+1)$. Therefore, the claim follows by induction on $n$. 
Inductively Defined Sets and Structural Induction
An inductive definition of a set $S$ has the following form:
(a) **Basis**: Specify one or more “initial” elements of $S$.
(b) **Induction**: Give one or more rules for constructing “new” elements of $S$ from “old” elements of $S$.
(c) **Closure**: The set $S$ consists of exactly the elements that can be obtained by starting with the initial elements of $S$ and applying the rules for constructing new elements of $S$.

The closure condition is usually omitted, since it is always assumed in inductive definitions.
Example

Let $S$ be the set defined as follows:

Basis: $0 \in S$

Induction: If $n \in S$, then $2n+1 \in S$.

Can you describe the set $S$?

$S = \{0,1,3,7,15,31,\ldots \} = \{ 2^n-1 \mid n \text{ a nonnegative integer} \}$

since $2^0-1=0$ and $2^{n+1}-1 = 2(2^n-1)+1$
We can define the set \( L \) of finite lists of integers as follows.

**Basis:** The empty list \( () \) is contained in \( L \)

**Induction:** If \( i \) is an integer, and \( l \) is a list in \( L \), then \( \text{(cons } i \ l \text{)} \) is in \( L \).

[Note: This is the Lisp style of lists, where \( \text{(cons } i \ l \text{)} \) appends the data item \( i \) at the front of the list \( l \)]

**Example:** \( \text{(cons } 1 \ (\text{cons } 2 \ (\text{cons } 3 \ () \text{)})} \) is the list \( (1 \ 2 \ 3) \) in Lisp.
We can define the set $B$ of binary trees over an alphabet $A$ as follows:

**Basis:** $\langle \rangle \in B$.

**Induction:** If $L, R \in B$ and $x \in A$, then $\langle L, x, R \rangle \in B$.

**Example:** $\langle \langle \rangle, 1, \langle \rangle \rangle$ // tree with one node (1)

**Example:** $\langle \langle \rangle, 1, \langle \rangle, r, \langle \rangle, 2, \langle \rangle \rangle$ // tree with root $r$ and two children (1 and 2).
In structural induction, the proof of the assertion that every element of an inductively defined set $S$ has a certain property $P$ proceeds by showing that

Basis: Every element in the basis of the definition of $S$ satisfies the property $P$.

Induction: Assuming that every argument of a constructor has property $P$, show that the constructed element has the property $P$. 
Example: Binary Trees

Recall that the set $B$ of binary trees over an alphabet $A$ is defined as follows:

Basis: $\langle \rangle \in B$.

Induction: If $L, R \in B$ and $x \in A$, then $\langle L, x, R \rangle \in B$.

We can now prove that every binary tree has a property $P$ by arguing that

Basis: $P(\langle \rangle)$ is true.

Induction: For all binary trees $L$ and $R$ and $x$ in $A$, if $P(L)$ and $P(R)$, then $P(\langle L, x, R \rangle)$.
Example: Binary Trees

Let $f: B \rightarrow \mathbb{N}$ be the function defined by

$$f(\langle \rangle) = 0$$

$$f(\langle L, x, R \rangle) = \begin{cases} 1 & \text{if } L = R = \langle \rangle \\ f(L) + f(R) & \text{otherwise} \end{cases}$$

**Theorem:** Let $T$ be a binary tree. Then $f(T)$ yields the number of leaves of $T$. 

25
**Example: Binary Trees**

**Theorem:** Let $T$ in $B$ be a binary tree. Then $f(T)$ yields the number of leaves of $T$.

**Proof:** By structural induction on $T$.

**Basis:** The empty tree has no leaves, so $f(\langle \rangle) = 0$ is correct.

**Induction:** Let $L, R$ be trees in $B$, $x$ in $A$.

Suppose that $f(L)$ and $f(R)$ denotes the number of leaves of $L$ and $R$, respectively.

If $L = R = \langle \rangle$, then $\langle L, x, R \rangle = \langle \langle \rangle, x, \langle \rangle \rangle$ has one leaf, namely $x$, so $f(\langle \langle \rangle, x, \langle \rangle \rangle) = 1$ is correct.
Example: Binary Trees

If L and R are not both empty, then the number of leaves of the tree \( \langle L, x, R \rangle \) is equal to the number of leaves of L plus the number of leaves of R. Hence, by induction hypothesis, we get

\[
f(\langle L, x, R \rangle) = f(R) + f(L)
\]

as claimed.

This completes the proof.
Counting
Counting Rules and Principles

- Product rule
- Sum rule
- Subtraction rule, principle of inclusion-exclusion
- Pigeonhole principle
- Permutations and combinations
- Binomial theorem and identities
- Pascal’s identity and Pascal’s triangle
Modeling and Solving Recurrences
Tower of Hanoi (1/3)

- Initially, n discs are placed on the first peg. Move the n discs one at a time from one peg to another such that no larger disc is ever placed on a smaller disc.

- Goal: Move the discs from peg 1 to peg 2.

- Let $H_n$ denote the number of moves needed to solve the tower of Hanoi problem with n discs.

- The steps are:
  1. Move the top n-1 discs from peg 1 to peg 3 using $H_{n-1}$ moves.
  2. Move the largest disc from peg 1 to peg 2.
  3. Move the n-1 discs from peg 3 to peg 2 using $H_{n-1}$ moves.
The steps are:

1. Move the top n-1 discs from peg 1 to peg 3 using $H_{n-1}$ moves.
2. Move the largest disc from peg 1 to peg 2.
3. Move the n-1 discs from peg 3 to peg 2 using $H_{n-1}$ moves.

Thus, we have the recurrence for $H_n$ as follows.

Initial condition: $H_1 = 1$

Recurrence relation: $H_n = 2H_{n-1} + 1$ for $n \geq 2$

For small values of $n$, we get $(H_1, H_2, H_3, H_4,...) = (1,3,7,15,...)$

Therefore, we can guess that $H_n = 2^n - 1$
How can we prove that $H_n = 2^n - 1$ holds for all $n \geq 1$?

By induction.

Base Step: $H_1 = 1 = 2^1 - 1$, so our claim holds for $n=1$.

Inductive Step: Suppose that $H_n = 2^n - 1$ holds for some $n \geq 1$. It follows that

$$H_{n+1} = 2H_n + 1 = 2(2^n - 1) + 1 = 2^{n+1} - 2 + 1 = 2^{n+1} - 1$$

Therefore, the claim follows by induction on $n$. 
A young pair of rabbits, one of each sex, is placed on an island. A pair of rabbits does not breed until they are 2 months old. After they are 2 months old, each pair of rabbits produces another pair each month.

Assume that none of the rabbits die.

How many pair of rabbits are there after n months?
Let $f_n$ denote the number of pairs of rabbits after $n$ months.

$f_1 = 1 \{\text{reproducing pairs} = 0, \text{young pairs} = 1\}$

$f_2 = 1 \{\text{reproducing pairs} = 0, \text{young pairs} = 1\}$

$f_3 = 2 \{\text{reproducing pairs} = 1, \text{young pairs} = 1\}$

$f_4 = 3 \{\text{reproducing pairs} = 1, \text{young pairs} = 2\}$

$f_5 = 5 \{\text{reproducing pairs} = 2, \text{young pairs} = 3\}$
The rabbit population can be modeled by a recurrence relation.

At the end of the first month, the number of pairs of rabbits on the island is \( f_1 = 1 \).

At the end of the second month, the number of pairs of rabbits on the island is \( f_2 = 1 \).

The number of pairs of rabbits after \( n \) months \( f_n \) is equal to the number of pairs of rabbits from the previous month \( f_{n-1} \) plus the number of pairs of newborn rabbits, which equals \( f_{n-2} \), since each newborn pair comes from a pair that is at least two months old, so \( f_n = f_{n-1} + f_{n-2} \) for \( n \geq 3 \).
Generating Functions

Given a sequence \((a_0, a_1, a_2, a_3, \ldots)\) of real numbers, one can form its generating function, an infinite series given by

\[ \sum_{k=0}^{\infty} a_k x^k \]

The generating functions is a formal power series, meaning that we treat it as an algebraic object, and we are not concerned with convergence questions of the power series.
Sum of Sequences

Let \((a_k)_{k \geq 0}\) and \((b_k)_{k \geq 0}\) be sequences with generating functions \(a(x)\) and \(b(x)\), respectively.

Then \((a_k + b_k)_{k \geq 0}\) has the generating function

\[ a(x) + b(x). \]
Let $a(x) = \sum_{k=0}^{\infty} a^k x^k$ and $b(x) = \sum_{k=0}^{\infty} b^k x^k$.

Then

$$a(x)b(x) = \sum_{k=0}^{\infty} \left( \sum_{j=0}^{k} a_j b_{k-j} \right) x^k$$
Shifting Sequences

Let $G(x)$ be the generating function of the sequence $(a_k)_k$. Then

$$xG(x) = \sum_{k=0}^{\infty} a_k x^{k+1} = \sum_{k=1}^{\infty} a_{k-1} x^k.$$
Solving a Recurrence

Suppose we have the recurrence system with initial condition \( a_0 = 2 \) and recurrence \( a_k = 3a_{k-1} \) for \( k \geq 1 \).

\[
G(x) - 3xG(x) = \sum_{k=0}^{\infty} a_k x^k - 3 \sum_{k=1}^{\infty} a_{k-1} x^k \\
= a_0 + \sum_{k=1}^{\infty} (a_k - 3a_{k-1}) x^k \\
= 2.
\]

Thus, \( G(x) - 3xG(x) = (1 - 3x)G(x) = 2 \). Hence,

\[
G(x) = \frac{2}{1 - 3x}.
\]
Solving a Recurrence

Since we know that $1/(1-ax)=1+ax+a^2x^2+...,$
we have

$$G(x) = 2(1+3x+3^2x^2+...).$$

Therefore, a sequence solving the recurrence is given by

$$(2,2x,3,2x3^2,...)=(2\times3^k)_{k\geq0}$$