Review material for Midterm

- 9 Lecture notes from 4/4 to 4/25
- Chapters 1, 2, 3
  Sections 4.1, 4.2, 4.5
  Sections 16.1, 16.2, 16.4
- Problem sets 1, 2, 3 and 4
- Solutions of problem sets
Topics

1. Math and proof-technique review
2. Order of growth
3. Recurrence relations
   ▶ linear recurrences, divide-and-conquer recurrences.
   ▶ Explicit substitution for solving simple recurrence relations
   ▶ The master theorem/method for DC recurrences
4. Divide-and-conquer algorithms
5. Greedy algorithms
1. Math and proof-technique review

Math

1. Set notation
2. Set of functions
3. Summation – see Appendix A.1
   
   **Arithmetic series:** \( \sum_{i=1}^{n} i = 1 + 2 + \cdots + n = ? \)

   **Geometric series:** \( \sum_{i=0}^{n} x^i = 1 + x + \cdots + x^n = ? \)

   **Harmonic series:** \( \sum_{i=1}^{n} \frac{1}{i} = 1 + \frac{1}{2} \cdots + \frac{1}{n} = ? \)

4. Fibonacci numbers
5. Binomial coefficients
6. Floor and ceiling
7. Logarithm and exponential
8. L’Hôpital’s rule
1. Math and proof-technique review

Proof-techniques

Proof by

▶ Definition (constructive existence)
▶ Induction
▶ Contradiction
▶ ...

...
2. Order of Growth

- Describe behaviors of functions in the limit ...
- Asymptotic definitions (notations)
  - $O(g(n)) = \{ f(n) : \exists \text{const. } c, n_0 \text{ s.t. } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \}$
  - $\Omega(g(n)) = \{ f(n) : \exists \text{const. } c, n_0 \text{ s.t. } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0 \}$
  - $\Theta(g(n)) = \{ f(n) : \exists \text{const. } c_1, c_2, n_0, \text{ s.t. } 0 \leq c_1 g(n) \leq f(n) \leq c_2 n \text{ for all } n \geq n_0 \}$
- Proof by definition
- Order of growth for frequently used functions:
  \[ \lg n, \ldots n, \ldots, n^k, \ldots, 2^n \]
3. Recurrence relations

Types:

- Linear recurrences
  \[ T(n) = c_1 T(n-1) + \cdots + c_k T(n-k) + f(n) \]

- Divide-and-conquer recurrences:
  \[ T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n) \]
  where \( a \geq 1 \) and \( b > 1 \), and \( f(n) \geq 0 \).

Methods to find the solution of a recurrence relation

- Direct iteration/substitution for simple recurrences.
- The master theorem/method for DC recurrences
3. Recurrence relations

The master theorem for solving DC recurrences:

- **Case 1**: If \( n^{\log_b a} \) is polynomially larger than \( f(n) \), i.e.,
  \[
  \frac{n^{\log_b a}}{f(n)} = \Omega(n^\epsilon) \quad \text{for some const. } \epsilon > 0,
  \]
  then \( T(n) = \Theta(n^{\log_b a}) \)

- **Case 2**: If \( n^{\log_b a} \) and \( f(n) \) are on the same order, i.e.,
  \[
  \frac{f(n)}{n^{\log_b a}} = \Theta(1),
  \]
  then \( T(n) = \Theta(n^{\log_b a \lg n}) \)

- **Case 3**: If \( f(n) \) is polynomially greater than \( n^{\log_b a} \), i.e.,
  \[
  \frac{f(n)}{n^{\log_b a}} = \Omega(n^\epsilon) \quad \text{for some const. } \epsilon > 0
  \]
  and \( f(n) \) is regular, then \( T(n) = \Theta(f(n)) \)
4. Divide-and-conquer algorithms

Three-step:

- **Divide** the problem into a number of (independent) subproblems,
- **Conquer** the subproblems by solving them recursively,
- **Combine** the solutions to the subproblems into the solution of the original problems.
4. Divide-and-conquer algorithms

Examples:
- Merge sort
- Max and Min
- Search for $A[i] = i$ in an sorted array $A$
- Maximum subarray
- Strassen’s algorithm
- Closest pair in 1-D
- k-way merge problem
- Integer multiplication
5. Greedy algorithms

- A greedy algorithm always makes the choice that looks best at the moment, without regard for future consequence “take what you can get now” strategy.

- The proof of the greedy algorithm producing the solution of maximum size of compatible activities is based on the following two key properties:
  - **The greedy-choice property**
    a globally optimal solution can be arrived at by making a locally optimal (greedy) choice.
  - **The optimal substructure property**
    an optimal solution to the problem contains within it optimal solution to subprograms.

- Greedy algorithms do not always yield optimal solutions, but for many problems they do.
5. Greedy algorithms

Examples:

1. Activity selection problems
2. Job scheduling (homework 4)
3. Huffman coding
4. 0-1 Knapsack problem
5. Money-change problem