Midterm Exam 1

• Thursday Feb. 10 in class
• Covered material: 1st class → the class on Tuesday Feb. 8th
• Closed-book and closed-notes
• Do not forget to prepare your cheat sheet (a single-side letter-size paper)
Review for Midterm Exam 1 – Chapter 1

What is the algorithm?
  • a sequence of unambiguous instructions for solving a problem

Algorithm design process

Typical problems discussed in this class: sorting, searching, string processing, graph problems, combinatorial problems, geometric problems, and numerical problems

The same problem can be solved by different algorithms with different efficiency

Typical data structures – array, linked list, stack, queue, graph, (adjacency matrix/linked-list), tree, binary tree and set

Pseudocode
Review for Midterm Exam 1 – Chapter 1

Graph
- Loop v.s. cycle
- Complete graph
- Edges and vertices: \( 0 \leq |E| \leq |V| (|V| - 1) / 2 \)
- Adjacency list and adjacency matrix for directed/undirected graph

Tree
- Connected and acyclic graph
- \(|E| = |V| - 1\)
- Height of the tree: \( \left\lfloor \log_2 n \right\rfloor \leq h \leq n - 1 \)
Time efficiency (complexity) of an algorithm

What is the input size and basic operation?

Measured by a function of the input size -- best case, worst case, average case

The order of the growth and how to prove it
  • ‘Limit’ technique
  • Definition

Three important symbols – $O(n)$, $\Theta(n)$ and $\Omega(n)$

Typical efficiency (complexity) class – constant, $\log n$, linear, $n \log n$, square, cubic, exponential, factorial, ……
## Polynomial and non-polynomial Complexity

<table>
<thead>
<tr>
<th>Expression</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1$</td>
<td>constant</td>
</tr>
<tr>
<td>$\log n$</td>
<td>logarithmic</td>
</tr>
<tr>
<td>$n$</td>
<td>linear</td>
</tr>
<tr>
<td>$n \log n$</td>
<td>$n \log n$</td>
</tr>
<tr>
<td>$n^2$</td>
<td>quadratic</td>
</tr>
<tr>
<td>$n^3$</td>
<td>cubic</td>
</tr>
<tr>
<td>$2^n$</td>
<td>exponential</td>
</tr>
<tr>
<td>$n!$</td>
<td>factorial</td>
</tr>
</tbody>
</table>
Review for Midterm Exam 1 – Chapter 2

Analyze the efficiency of nonrecursive algorithms

• Find all the loops
• The operation in the innermost loop is the basic operation
• Write the complexity in the form of summations
• Simplify the expression using formulas in Appendix A

Analyze the efficiency of recursive algorithms

• Find the recurrence relations and initial conditions
• Find the closed-form solution (Appendix B)
  – Forward substitution
  – Backward substitution
  – Linear 2\textsuperscript{nd} order with constant coefficients (homogenous and inhomogenous cases)
  – Properties of smooth functions
• Typical kinds of recurrence relations
• Master Theorem
Important Recurrence Types

One (constant) operation reduces problem size by one.

\[ T(n) = T(n-1) + c \quad T(1) = d \]
Solution: \[ T(n) = (n-1)c + d \] \( \text{linear, e.g., factorial} \)

A pass through input reduces problem size by one.

\[ T(n) = T(n-1) + cn \quad T(1) = d \]
Solution: \[ T(n) = \left[ n(n+1)/2 - 1 \right] c + d \] \( \text{quadratic, e.g., insertion sort} \)

One (constant) operation reduces problem size by half.

\[ T(n) = T(n/2) + c \quad T(1) = d \]
Solution: \[ T(n) = c \log_2 n + d \] \( \text{logarithmic, e.g., binary search} \)

A pass through input reduces problem size by half.

\[ T(n) = 2T(n/2) + cn \quad T(1) = d \]
Solution: \[ T(n) = cn \log_2 n + d n \] \( n \log_2 n, \text{e.g., mergesort} \)
Useful Formulas in Appendix A

Make sure to be familiar with them

\[
\sum_{i=startInd}^{endInd} 1 = 1 + 1 + \cdots + 1 = endInd - startInd + 1
\]

\[
\sum_{i=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n + 1)}{2} \in \Theta(n^2)
\]

\[
\sum_{i=1}^{n} i^k \in \Theta(n^{k+1})
\]

\[
\sum_{i=1}^{n} c a_i = c \sum_{i=1}^{n} a_i
\]

\[
\sum_{i=1}^{n} (a_i + b_i) = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i
\]

Prove by Induction

Note: this is not a full list in Appendix A! Copy the full list on your cheat sheet!
Three Recurrence Types We know How to Find the Closed-Form Solution

\[
T(n) = a \cdot T(n-1) + n^k
\]
\[
T(n) = a \cdot T(n/b) + n^k \quad (a \geq 1, b \geq 2) \Rightarrow \text{Master Theorem}
\]
\[
T(n) = a \cdot T(n-1) + b \cdot T(n-2) \Rightarrow \text{Linear Second Order}
\]

Please related them to the following algorithms we learned in the last class

• Recursive algorithm for computing \( n! \)
• Recursive algorithm for Tower of Hanoi
• Recursive algorithm for finding the number of digits in the binary representation of a decimal integer
• Recursive algorithm for finding the Fibonacci numbers
Three Recurrence Types We know How to Find the Closed-Form Solution

\[ T(n) = a \cdot T(n - 1) + n^k \]
\[ T(n) = a \cdot T(n/b) + n^k \quad (a \geq 1, b \geq 2) \Rightarrow \text{Master Theorem} \]
\[ T(n) = a \cdot T(n - 1) + b \cdot T(n - 2) \Rightarrow \text{Linear Second Order} \]

Forward substitutions (not recommended for complex patterns)

Backward substitutions (a general approach to solve recurrence, but not recommended for linear second order recurrence)

Linear 2\(^{nd}\) order recurrences with constant coefficients

The solution to important recurrence type (pay attention to the initial condition)

Master Theorem (recommended for solving general divide-and-conquer recurrence)