Assignment Info

• A2 marks out this morning
Exam Info

• All materials covered except for MDP
• You can bring: 1 A4 page, hand-written on both side
• 7 November, 9am, 2 hours, 7-11 Barry Drive
  • Please recheck time & location in ANU timetable
• Tips:
  • You will have 15 min to read the questions first. Use this time to roughly understand the question, see the points, and strategise which questions you’ll do first
  • Do questions with higher points and you can do them first
  • Try to answer all questions. If you have some writing even if it’s wrong, we’ll give 10% for writing marks. If it’s empty, it’s 0
To help prepare for exam

• This Wed, 23 Oct, usual Q&A session (Coombs): Recap of the materials
• Wed, 30 Oct, 4pm-5pm, in Copland Lecture Theatre: We’ll cover Part A of assignments
• Friday, 1 Nov, 10am-12am, in Copland Lecture Theatre: Drop in session, a couple of tutors would be available
COMP3600/6466 – Algorithms Recap

Hanna Kurniawati

https://cs.anu.edu.au/courses/comp3600/
What we’ve covered

• The problem: Setting Up the Stage
• Analysis of Algorithms
• Algorithm Design Techniques
• Complexity Classes
Setting Up the Stage

• Algorithms: What Is It
• Problem Definition for:
  • Sorting
  • Searching
• Model of Computation
What we’ve covered

 ✓ The problem: Setting Up the Stage
   • Analysis of Algorithms
   • Algorithm Design Techniques
   • Complexity Classes
Analysis of Algorithms

• Asymptotic Notations
• Divide-and-Conquer and Recurrence Analysis
• Randomized Algorithm and Probabilistic Analysis
• Empirical Analysis (not in exam)
• Correctness
## Asymptotic Analysis

<table>
<thead>
<tr>
<th>Asymptotic bounds of $f(n)$</th>
<th>Upper bound</th>
<th>Lower bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>May be tight or not</td>
<td>$O(g(n))$</td>
<td>$\Omega(g(n))$</td>
</tr>
<tr>
<td>Non-tight</td>
<td>$o(g(n))$</td>
<td>$\omega(g(n))$</td>
</tr>
<tr>
<td>Tight</td>
<td>$\Theta(g(n))$</td>
<td></td>
</tr>
</tbody>
</table>

- These notations are essentially sets of functions
- Membership test:
  - Finding the appropriate constants
  - Limit definition
Recurrence Analysis

• What is recurrence?
• Where is it used?
• Substitution Method: Solving recurrences
• Recursion Tree: Guessing the solution form
• Master’s Theorem: Finding asymptotic bound of a recurrence
Randomized Alg + Probabilistic Analysis

• A bit about Randomized Algorithm
• A bit about Probabilistic Analysis
• Example: Probabilistic Analysis of A Deterministic Algorithm (Insertion Sort)
• Example: Probabilistic Analysis of A Randomized Algorithm (Quick Sort)
Correctness

• Loop Invariant

• Other common proving techniques, such as: Proof by contradiction and direct proof, can be used too
What we’ve covered

✓ The problem: Setting Up the Stage
✓ Analysis of Algorithms
  • Algorithm Design Techniques
  • Complexity Classes
Algorithm Design Techniques

• Brute Force: When stuck, start here and improve
• Divide & Conquer: Discussed a bit in recurrence analysis
• Randomized Algorithm: Discussed a bit in prob. analysis
• Decrease & Conquer: Very similar to divide & conquer, but reduce to 1 smaller sub-problem (e.g., Insertion Sort)
• Transform & Conquer
• Dynamic Programming
• Greedy
• Additional Abstract Data Structures: Hash tables
Transform & Conquer

- **Instance Simplification**
  - Common method: Pre-sorting
    - Lower bound time complexity for comparison-based sorting
    - Non-comparison based sorting,
      - Rely on properties of the data
      - Counting Sort, Radix Sort, Bucket Sort

- **Representation Change:**
  - Various abstract data structures:
    - Binary Search Tree, AVL Tree, Red-Black Tree, Heaps
    - Evaluating and multiplying polynomials

- **Problem Reduction:** lcm and counting paths in a graph
Dynamic Programming (DP)

• Memory + reuse

• Two Requirements
  • Optimal substructure
  • Overlapping substructure

• Two types:
  • Top down
  • Bottom up

• Examples: Fibonacci, LCS

• DP in Optimization: MDP (not in exam)
Greedy

• Use locally optimal choice

• Requirements:
  • Optimal substructure
  • Greedy choice property

• Minimum Spanning Tree
Additional Data Structures: Hash Table

• What is a hash table?
• Hashing with Chaining
• Hashing functions:
  • Simple uniform hashing
  • Universal hashing
• Open addressing
• Perfect hashing
What we’ve covered

- The problem: Setting Up the Stage
- Analysis of Algorithms
- Algorithm Design Techniques
- Complexity Classes
Complexity Class

• Complexity classes: P & NP
• Showing a problem is P, NP, NP-complete
• Other complexity classes
What we’ve covered

✓ The problem: Setting Up the Stage
✓ Analysis of Algorithms
✓ Algorithm Design Techniques
✓ Complexity Classes
That’s all folks!!!