Algorithms and Data Structures 2
Exam Notes
Week 1: Divide-and-Conquer

Mads Richardt

Objective

This document is a self-contained set of notes and worked solutions for the **Week 1: Divide-and-Conquer** problem set. It contains:

- General methodology for solving recurrence relations.
- Relevant mathematical tools (geometric series, logarithm rules).
- Notes and derivations from the lecture slides.
- Complete solutions to the weekly problem set, with references back to the notes.

1 General Methodology

1.1 Recursion Tree Method

To solve a recurrence of the form

$$T(n) \leq aT\left(\frac{n}{b}\right) + f(n),$$

we:

1. Draw the recursion tree. The root is $T(n)$.
2. Expand each node into a children of size n/b.
3. Compute the work per level: sum of all $f(\cdot)$ terms.
4. Compute the number of levels: the depth is $\log_b n$ (until the subproblem size becomes constant).
5. Multiply out to find total work: sum of work over all levels.

Number of leaves: Each level multiplies the number of nodes by a. After $\log_b n$ levels, there are

$$a^{\log_b n} = n^{\log_b a}$$

leaves.

1.2 Substitution Method

To solve a recurrence, we:

1. Guess a bound $T(n) \leq cn \log n$ or similar.
2. Use induction to prove the bound holds.
3. Verify the base case.

1.3 Notation

- $O(g(n))$: upper bound.
- $\Omega(g(n))$: lower bound.
- $\Theta(g(n))$: tight bound.

2 Relevant Mathematical Tools

2.1 Geometric Series

For $x \neq 1$:

$$\sum_{i=0}^{m} x^i = \frac{x^{m+1}-1}{x-1}.$$

Special case $x = \frac{1}{2}$:

$$\sum_{i=0}^{\log n} \frac{1}{2^i} \leq 2.$$

2.2 Logarithm Rules

$$\log_b n = \frac{\log n}{\log b}, \quad a^{\log_b n} = n^{\log_b a}.$$

3 Notes from Slides

- Divide-and-conquer structure: divide, conquer, combine.
- Mergesort recurrence:
  $$T(n) \leq 2T(n/2) + cn.$$
  Solution: $T(n) \in \Theta(n \log n)$.
- Counting inversions: use divide-and-conquer with merge step that counts cross-inversions.

4 Solutions to Problem Set

1.1 Recurrence

$$T(n) \leq 2T(n/4) + cn.$$

Recursion Tree: At level i, there are 2^i nodes, each of size $n/4^i$. Work per node: $c \cdot (n/4^i)$. Work per level: $2^i \cdot c(n/4^i) = c \cdot n \cdot (1/2)^i$.

$$\text{Total work} = cn \sum_{i=0}^{\log_4 n} \left(\frac{1}{2}\right)^i \leq 2cn.$$

$$\Rightarrow T(n) \in \Theta(n).$$

Substitution: Guess $T(n) \leq kn$. Induction step:

$$T(n) \leq 2k(n/4) + cn = \frac{k}{2}n + cn.$$

For $k \geq 2c$, $T(n) \leq kn$.

$$\Rightarrow T(n) \in O(n).$$

1.2 Recurrence

$$T(n) \leq 2T(n/4) + c\sqrt{n}.$$

Recursion Tree: Work per level: $2^i \cdot c\sqrt{n/4^i} = c\sqrt{n} \cdot (2^i/2^i) = c\sqrt{n}$. Number of levels: $\log_4 n$.

$$\text{Total work} = c\sqrt{n} \cdot \log_4 n.$$

$$\Rightarrow T(n) \in \Theta(\sqrt{n}\log n).$$

2. Counting Inversions (KT 4.2)

Divide-and-conquer:

1. Split array into halves.
2. Recursively count inversions in each half.
3. Count cross-inversions during merge: whenever $a[i] > b[j]$, add $(\text{mid} - i + 1)$ inversions.

$$T(n) \leq 2T(n/2) + cn \quad \Rightarrow \quad T(n) \in \Theta(n \log n).$$

3. Divide-and-Conquer on Trees (KT 4.6)

- Use recursion: pick the middle node, recursively process subtrees.
- Time: $O(n)$.

4. Divide-and-Conquer on Grid Graphs (KT 4.7)

- Divide the grid into quadrants.
- Recurse on subgrids, combine at boundaries.
- Recurrence depends on structure, usually $O(n \log n)$.

5. CSES Exercises

Missing Number (1083): Sum formula vs. array sum: $O(n)$.
Distinct Numbers (1621): Sort array, count distinct: $O(n \log n)$.

6.1 Recurrence

$$T(n) \leq T(3n/4) + cn.$$

Recursion tree: work per level decreases linearly, depth $\log_{4/3} n$.

$$\Rightarrow T(n) \in \Theta(n).$$

6.2 Recurrence

$$T(n) \leq T(n/2) + T(n/3) + T(n/6) + cn.$$

By expansion, the tree has total branching factor summing to 1.

$$\Rightarrow T(n) \in \Theta(n \log n).$$

7. Divide-and-Conquer on Trees II

If tree is unbalanced: use separator lemma. Always possible to find a node splitting tree into subtrees of size $\leq n/2$.

$$\Rightarrow O(\log n) \text{ probes.}$$

Summary

- Recurrences solved explicitly (with references to recursion trees and substitution).
- Inversions solved with divide-and-conquer.
- Trees and grid graphs analyzed.
- CSES tasks solved.