Slides for a Course on the Analysis and Design of Algorithms

Chapter 8: Problem Complexity

Stephen J. Hegner
Department of Computing Science
Umeå University
Sweden
hegner@cs.umu.se
http://www.cs.umu.se/~hegner

8. Problem Complexity

- In the previous part of the course, the focus has been upon the complexity of specific algorithms.
- In this section, the question of identifying lower bounds on the complexity for all algorithms which solve a given problem will be studied.
- At this point, attention will be restricted to a single but very important problem: sorting.

8.1 The complexity of sorting

8.1.1 Decision trees

- The following assumptions define the problem context:

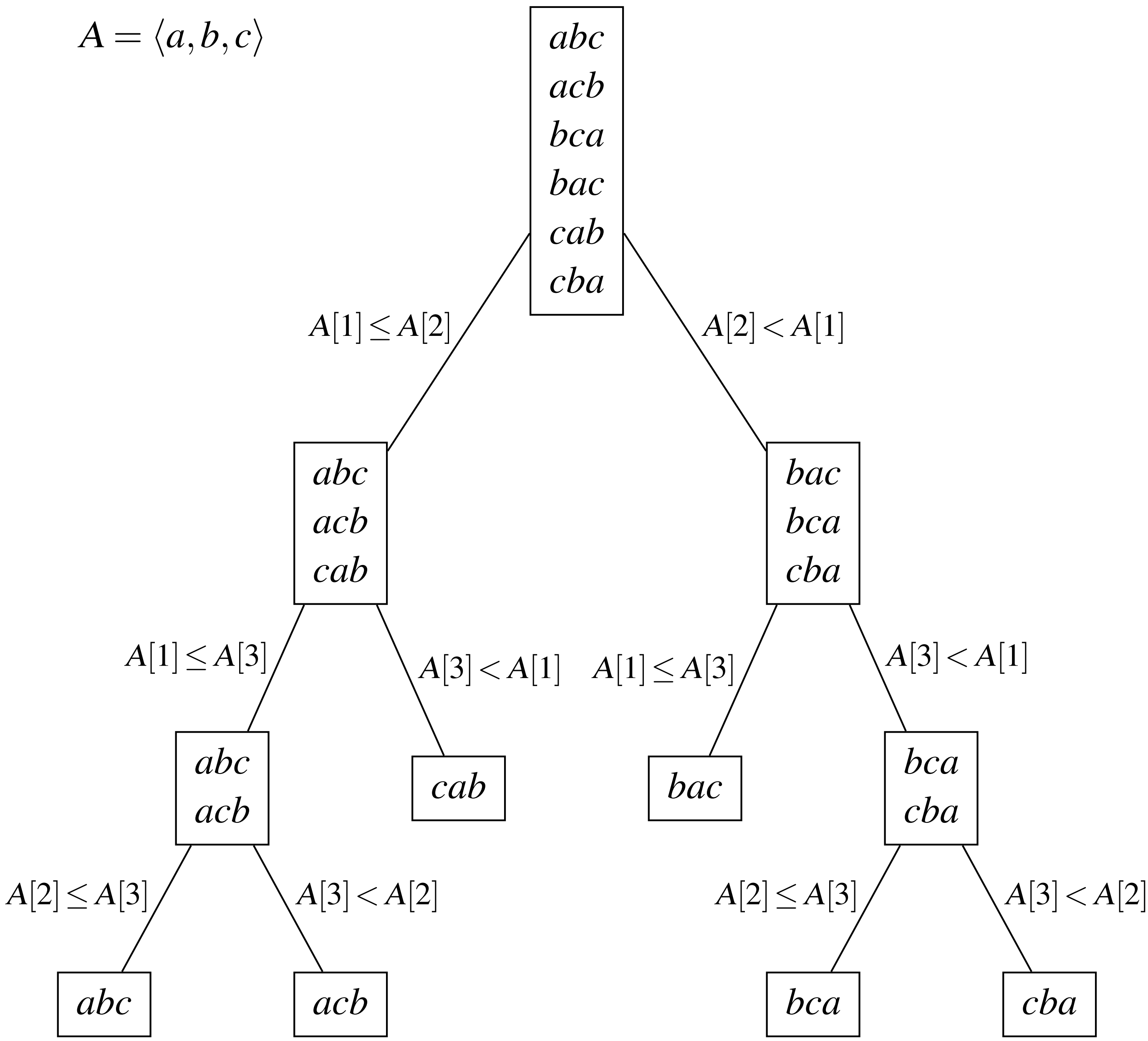
  (i) Given is an array of elements of data type S:

  $$A: \ \text{array}[1..n] \text{ of } S;$$

  (ii) S has a total order $\leq$, but no other operations.

  (iii) Only comparisons of elements provide information about the associated irreflexive ordering $<$ on S.

- A convenient means of representing the sorting process is via *decision trees*.
- The decision tree for $S = \{a, b, c\}$ with $A = \langle a, b, c \rangle$ is shown on the next page.



- The above convention preserves the initial order of equal elements (*stable sort*).
- Note that there are six leaves, one for each permutation of the array.
- In general, for an array of n elements, there will be $n!$ leaves to the decision tree.

8.1.2 Terminology Let T be a binary tree.

- Recall that $\mathsf{EPL}(T)$ denotes the (total) *external path length*, and $\mathsf{LeafNode}(T)$ denotes the number of leaf vertices. (See 2.2.5.)

(a) The average external path length of T, denoted $\mathsf{AvgExtPL}(T)$, is $\mathsf{EPL}(T)/\mathsf{LeafNode}(T)$.

8.1.3 Lemma *For any binary tree T,*

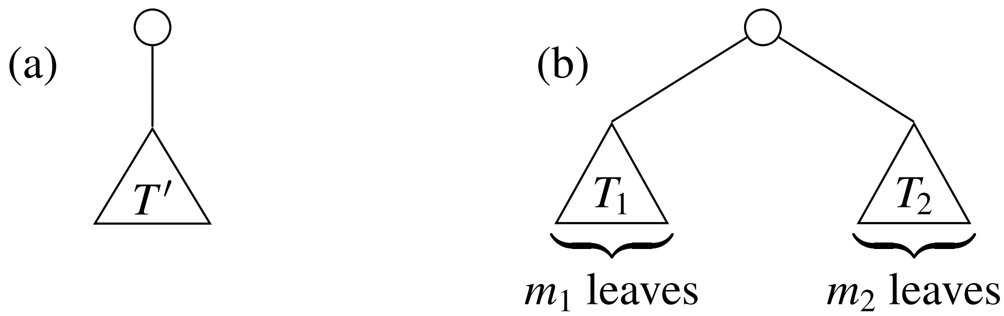
$$\mathsf{AvgExtPL}(T) \geq \log_2(\mathsf{LeafNode}(T))$$

PROOF:

- The proof is by induction on the length d of the longest path from the root to a leaf.

Basis: For $d = 0$, the proof is trivial.

Step: Assume that the assertion is true for all path lengths $c < d$, and let T' be a binary tree whose longest external path has length d. There are two possibilities for the shape of this tree:



Case (a) is trivial. For case (b), note that T_1 and T_2 each satisfy the inductive hypothesis, since they are of height at most $d-1$, and:

$$
\begin{aligned}
&\mathsf{AvgExtPL}(T) \\
&= (1+\mathsf{AvgExtPL}(T_1)) \cdot \left(\frac{m_1}{m_1+m_2}\right) + (1+\mathsf{AvgExtPL}(T_2)) \cdot \left(\frac{m_2}{m_1+m_2}\right) \\
&\geq 1+\log_2(m_1) \cdot \left(\frac{m_1}{m_1+m_2}\right) + \log_2(m_2) \cdot \left(\frac{m_2}{m_1+m_2}\right) \\
&= 1+\left(\frac{1}{m_1+m_2}\right) \cdot (m_1 \cdot \log_2(m_1) + m_2 \cdot \log_2(m_2))
\end{aligned}
$$

- The minimum of this value occurs when $m_1 = m_2$. (Replace m_2 with $m - m_1$ in the above formula, and differentiate with respect to m_1. The derivative is zero when $m_1 = m - m_1$, and it is easily verified that the second derivative is positive.)
- However, when $m_1 = m_2 = m/2$, the above formula becomes just $1 + \log_2(m)$. Thus, to complete the above line of inequalities, the following is added:

$$
\geq 1+\log_2(m)
$$

□

8.1.4 Theorem *Any comparison-based sorting algorithm must have average time complexity at least as great as* $\Theta(n \cdot \log(n))$*, with n the size of the list to be sorted.*

PROOF: This follows directly from the above lemma and *Stirling's approximation*, which states that

$$
\log(n!) = n \cdot \log(n) + f(n)
$$

with $f(n) \in O(n)$. □