1 Motivation

One major motivation of this lecture is to derive a lower bounding method for comparison-based sorting algorithms. In particular, we want to have a good lower bound on the number of comparisons. To that end, we introduce a generic lower bounding method, called the decision tree method. However, it should be understood that the decision tree method can go much beyond the context of either sorting algorithms or the bound on the number of comparisons.

2 Query-based algorithm abstraction

Let us focus on the deterministic algorithms (i.e., no randomness is used by the algorithm). Let us imagine a generic form of such algorithms as follows:

1. The algorithm \( A \) makes a query about the input. Here queries refer to questions with a small number of possible answers.
2. Based on the query, its answer and the previous history, the algorithm \( A \) will either choose to make another query (go back to Step (1)), or stop and output.

Remark: the above abstraction of algorithms does not characterize all possible forms of algorithms. However, it is good enough for our purpose.

For example, the queries in sorting algorithm could be asking whether \( x_i < x_j \) for some \( i \neq j \). The possible answers are yes and no!

Another example could be the queries in the binary search, which are simply whether \( k < x_i \) for some key \( k \) and some index \( i \). The possible answers are yes and no!

3 Decision Tree

We introduce the definition of decision trees here, which are imaginary trees for analysis purpose only. It is not a data-structure stored anywhere.

**Definition 1 (Decision Tree)** A decision tree is a tree such that

- Each internal node is labeled by a query, which is some question about the input.
- The edges out of a node correspond to the possible answers to the node’s query.
- Each external node (leaf) is labeled with an output.

Remark: A decision tree basically gives an algorithm \( A \) for the problem. Such an algorithm \( A \) just needs to make the queries and follows the answers from the root of the decision tree. When \( A \) reaches an external node in the decision tree, it outputs the content in the label of that node.

Examples: (1) the decision tree to choose one of six animals. (2) the decision tree for binary search. (3) the decision tree for sorting algorithms.
The decision tree for binary search

Let \( k \) be the key to search and \( n \) the number of items. The query in the root is whether \( k < x_{n/2} \), i.e., comparing \( k \) with the value stored in the middle. There could be two possible answers, yes and no. All the external nodes are labeled with either a specific position that stores value \( k \) or "no such key". Depending on the answer and previous queries and answers, the algorithm will make a new query or output the result accordingly.

The decision tree for sorting algorithms

Let \( n \) be the number of items. The query in the root could be any comparison, e.g., \( x_i < x_j \) for an arbitrary choice of \( i, j = 1, \cdots, n \). There could be two possible answers, yes and no\(^2\). All the external nodes are labeled with a specific sorting result that could be any permutation of the original \( n \) input items.

4 Properties about the decision tree

From the definition of the decision tree \( T \), we can have the following simple observations.

- The height of \( T \) is the number of queries made by the algorithm. Note this is not necessarily the running time of the algorithm, as the algorithm might spend a lot of time between each query.
- If the number of possible answers for each query is at most \( d \), then the decision tree is a \( d \)-ary tree. When \( d = 2 \), it is a binary tree.
- The number of external nodes is at least the number of possible outcomes of any algorithm on all possible inputs. This is because the algorithm outputs correctly on all possible inputs.

Thus, by the relationship between the number of external nodes \( e(n) \) and the height \( h \) of \( T \) (Theorem 2.8 on page 84), we have

\[
    h + 1 \leq e(n) \leq 2^h,
\]

which leads to a simple lower bound of \( h \) as follows,

\[
    h \geq \Omega(\log(e(n))).
\]

This is also the lower bound on the number of comparisons and the running time.

5 Application to binary search and the sorting algorithms

Now we are ready to see the applications of the decision tree method. The crux is to estimate \( e(n) \).

- **Binary Search:** in this case, the number of all possible outcomes is \( n + 1 \). (i.e., \( n \) positions or "no such key"), which implies that \( e(n) \geq n + 1 \). Thus, we have the lower bound of the number of comparisons and the running time,

\[
    h \geq \Omega(\log(e(n))) = \Omega(\log(n)).
\]

- **Sorting Algorithm:** the number of all possible outcomes is \( n! \). (i.e., all possible permutations of \( n \) items. To get the lower bound, we use the famous Stirling’s approximation for \( n! \) as follows.

  **Fact 2 (Stirling’s approximation)**

\[
    \ln(n!) = n \ln(n) - n + O(\ln(n)), \text{ for large } n.
\]

Thus, we have the following lower bound,

\[
    h \geq \Omega(\log(e(n))) = \Omega(\log(n!)) = \Omega(n \log(n)).
\]

---

\(^2\)As a simple extension, one can consider the query is to ask about the relationship between \( x_i \) and \( x_j \). In that case, there are three possible answers: equal, smaller, and larger.