Lower Bounds (See Lect Notes) [Ch 8, Baas-Van Gelder, D.E. Knuth vol 2]

This topic concerns lower bounds on problem complexity. They may be in the form of order notation: \( P \) has \( \Omega (\cdot) \) complexity or exact \( P \) has a lower bound of \((n-1)\). (always better)

Example 1. Sorting using Comparisons: (Ch 8)

Decision Tree Approach:

Algorithms that use only comparisons, will start with some pair first; depending on the result will do some pair next and so. Corresponding to an algorithm, there is a decision tree whose leaf nodes correspond to permutations and this is a tertiary tree in general. If we assume that given input has no two equal values, the tree is binary (such cases give a lower bound without loss). The greatest depth from the root to a leaf node (called height of tree) gives us the \( \Omega \) complexity of the algorithm. Now we illustrate this with a small size example: \( n = 3 \).
Since there are $n!$ leaf nodes on this tree for each algorithm, and height $h$ satisfies

$$2^h \geq n!$$

$$\therefore h \geq \log(n!) = \Omega(n \log n)$$

$$\rightarrow \Theta(n \log n).$$

Hence the worst case complexity of an algorithm to sort using comparisons (only) is $\Omega(n \log n)$.

Example 2: Input: Unsorted array $A[1...n]$ of Numbers

Desired output: Maximum element (one such choice if there are ties)

Lower Bound on W.C. Complexity: \( n-1 \).

**Pf:** Consider an array of distinct elements.

In each comparison, larger element "wins" and the other "loses".

In each comparison, exactly one element "loses".

Any correct algorithm (including those that perform "dumb" operations) needs to eliminate all except one element.

\[ \# \text{ of comparisons needed} \geq n-1 \text{ in W.C.} \]

---

**Adversary Argument Approach**

Now we revisit one of the first problems discussed in D. and C algorithms: MAX-MIN.

There we presented algorithms whose W.C. Complexity was \( \frac{3}{2} n - 27 \). We had asked a question at that time: Is there an algorithm whose worst-case complexity is lower? Yes \( \rightarrow \) provide algorithm. No \( \rightarrow \) provide a proof.

We now show that answer is No using an adversary strategy.
Imagine the following scenario:

There are two "players" (people)

Player I: Does not know the input; but designs the algorithm (has control over which step are done under various situations)

Player II: Does not know (or control) what steps will be taken by & Player I. But has complete control over the input (and hence how alg. designer's questions are answered subject then being some input for which II's answers are valid)

Adversary Strategy:

A set of answers for all possible questions of alg. designer at all possible situations for which there is at least one input.

Adversary wants algorithm designer to take more steps and alg. designer wants the opposite.

Both players know all of the above.
Now we focus on **MAX-MIN**.

Description of Adversary Strategy etc

Adversary "classify" elements in the array into 4 groups at any stage.

N: elements that have not been used so far in any comparison.

W: elements that have "won" at least once in previous comparisons but never "lost" in them.

L: elements that have "lost" at least once but never won in previous comparisons.

WL: elements that have "won" at least once and "lost" at least once.

Elements in W **cannot** be MIN (one unit of info)

L **cannot** be MAX (""

WL **cannot** be either (2 units of info)

N: 0 units of info.
At each step, total info. content is non-decreasing as the algorithm progresses for any algorithm.

In one comparison, we can either get 0, 1 or 2 units of information.

Either an element "wins" for the first time (i.e. you have eliminated this from being MIN)

\[ \text{info gain} = 1 \]

\[ \text{info gain} = 1 \] or "loses" for the first time (i.e. you have eliminated this from being MAX)

\[ \text{info gain} = 0 \] or one element "wins" for first time and other "loses"

\[ \text{info gain} = 2 \] or none of the above

\[ \text{info gain} = 0 \]
Every valid algorithm has its content go up from 0 to \((2n-2)\).

Algorithm designer wants as few steps as possible adversary given result so that alg. designer is "forced" to take as many steps as possible but all his/her answers are valid.

This is what "Adversary Strategy" is all about. Now the description for MIN-MAX problem that "forces" alg. designer to take at least \(\lceil \frac{3}{2} n - 2 \rceil\) steps.

At any stage the process, adversary needs to answer the result of comparison of two types

\[ A_{[c]} = x : y = A_{[j]} \]

For this, adversary bases his answers on the types of \(A_{[c]}\) and \(A_{[j]}\). Each of these could be in one of four types: \(W, L, WL, N\).

So we get sixteen possible cases for each of which adversary needs to decide how to answer. [Of course answers depend on type]
But before, deciding adv. answers, let us look at possible consequence, in terms of information gain for alg. designer for each of these combinations.

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>W</th>
<th>L</th>
<th>WL</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>2</td>
<td>½</td>
<td>½</td>
<td>1</td>
</tr>
<tr>
<td>W</td>
<td>½</td>
<td>1</td>
<td>½</td>
<td>≤1</td>
</tr>
<tr>
<td>L</td>
<td>½</td>
<td>0</td>
<td>½</td>
<td>≤1</td>
</tr>
<tr>
<td>WL</td>
<td>1</td>
<td>≤1</td>
<td>≤1</td>
<td>0</td>
</tr>
</tbody>
</table>

Black: fixed info gain no matter how adv. answers
Blue: upper limit on info gain no matter how adv. answers
Green: Min info gain
Red: Max info gain depends on adv. answers.

Adv. Strategy: avoid 2 if you can.

Adversary's answers to achieve the above:

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>W</th>
<th>L</th>
<th>WL</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>*</td>
<td>x&lt;y</td>
<td>x&gt;y</td>
<td>x</td>
</tr>
<tr>
<td>W</td>
<td>x&gt;y</td>
<td>*</td>
<td>x&gt;y</td>
<td>**</td>
</tr>
<tr>
<td>L</td>
<td>x&lt;y</td>
<td>x&lt;y</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>WL</td>
<td>*</td>
<td>**</td>
<td>**</td>
<td>*</td>
</tr>
</tbody>
</table>

* : be consistent with previous answers
** : be consistent with previous answers

These answers have no difficulty of being consistent with previous answers and avoiding giving 2 units when possible.
We have described adversary's book of instruction to "force" alg. designers to use $\sqrt{\frac{3}{2}n-2}$ comparisons. We show this next.

Suppose in some valid algorithm, there are

$k : N-N \text{ Comparisons ; info gain } \leq 2$
$l : \text{ other } \quad ; \quad l \leq 1$

\[ \text{ total # of Comparisons in this algorithm } = k + l. \]

But since an element belongs to set $N$ only until it is used for the first time,

\[ k \leq \left\lfloor \frac{n}{2} \right\rfloor. \]

Total information gained in the algorithm

\[ \leq 2k + l. \]

The algorithm needs, in order to be valid,

$(2n-2)$ units of information.

\[ (2k+l) \geq 2n-2 \]

\[ (k+l) \geq 2n-2-k \]

\[ \geq 2n-2 - \left\lfloor \frac{n}{2} \right\rfloor \]

\[ = \left\lfloor \frac{3}{2}n-2 \right\rfloor. \]

Hence, this is a valid lower bound.