Computer Science 385
Design and Analysis of Algorithms
Siena College
Spring 2025

Algorithm Limitations Practice

Lower Bounds

A *lower bound* gives

An established lower bound means

Lower bounds can be exact but are often expressed as a Big-______ efficiency class.

A lower bound is *tight*

Lower Bound Examples

- the number of comparisons needed to find the largest element in an unordered set of n numbers

  Lower bound: $\Omega(\qquad)$ tight?

- number of comparisons needed to sort an arbitrary array of size n

  Lower bound: $\Omega(\qquad)$ tight?

- number of comparisons necessary for searching in a sorted array of n numbers

  Lower bound: $\Omega(\qquad)$ tight?

- the number of comparisons needed to determine if all elements of an array of n elements are unique

  Lower bound: $\Omega(\qquad)$ tight?

- number of steps needed to multiply two n-digit integers
  Lower bound: $\Omega(\qquad)$ tight?
- number of multiplications needed to multiply two $n \times n$ matrices
  Lower bound: $\Omega(\qquad)$ tight?

Trivial Lower Bounds

A *trivial lower bound* is established by

Generating all permutations of a set of n items

Trivial lower bound: $\Omega(\qquad)$ Tight bound?

Why?

Evaluating a polynomial of degree n, $p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$

Trivial lower bound: $\Omega(\qquad)$ Tight bound?

Why?

Multiply two $n \times n$ matrices

Trivial lower bound: $\Omega(\qquad)$ Tight bound?

Why?

Traveling Salesman with n cities

Trivial lower bound: $\Omega(\qquad)$ Tight bound?

Why?

What about finding an element in a collection of size n?

Information Theoretic Lower Bounds

An *information-theoretic lower bound* is based on

Binary search in a sorted array of size n

A Decision Tree: Find minimum of three numbers a, b, c

Decision tree properties

- The number of leaves must be
- The operation of an algorithm on a particular input is modeled by
- The number of comparisons is equal to
- Worst-case behavior is determined by

Any such tree with a total of l leaves (outcomes) must have $h \geq$

Decision Tree for Selection Sort of 3 elements

Decision Tree for Insertion Sort of 3 elements

Any **comparison-based** sorting algorithm can be represented by a decision tree. Assuming our input has n values:

- The number of leaves (outcomes) must be $\geq$

- The height of binary tree with ____ leaves $\geq$

- This tells us the number of comparisons in the worst case
  $C_{worst}(n) \geq$

  for **any** comparison-based sorting algorithm. (!!)
- Is this bound tight?

The idea of *adversary arguments* to find a lower bound is discussed in the notes.

Problem Reduction

Idea: solving a problem by using an existing solution to another problem.

An example from programming:

You wish to draw a circle, and already have a method to draw an ellipse:

```
draw_ellipse(double horiz, double vert, double x, double y)
```

So we use it in our implementation of

```
draw_circle(double radius, double x, double y) {





}
```

We have *transformed* or *reduced* the problem of drawing a circle to the problem of drawing an ellipse.

We have "proven" that drawing a circle is ________________________________ drawing an ellipse.

Example 1: The Pairing Problem

Problem statement: given two n-element arrays $A1$ and $A2$. Your task is to rearrange the values in $A2$ such that the smallest value in $A2$ is paired with the smallest value in $A1$. The second smallest in $A2$ is paired with the second smallest in $A1$, and so one. Only values of $A2$ are rearranged; $A1$ is unchanged.

Example Input:

$A1$	23	5	57	45
$A2$	150	175	100	120

Output:

$A1$				
$A2$				

Assume we have some algorithm that can solve this.

Now consider the sorting problem for an n-element array, $A[0..n-1]$. How can we use the solution to the pairing problem to solve the sorting problem?

What is the total cost of this approach?

We have *reduced* the sorting problem to an instance of the pairing problem.

Using problem reduction to show a lower bound

- If problem A is **at least as hard as** problem B, then a lower bound for B is also a lower bound for A.
- Hence, we wish to find a problem B with a known lower bound that can be reduced to the problem A.

In our example, problem A is the ____________________ problem

problem B is the ____________________ problem

Problem B (the ____________________ problem) has a known lower bound of $\Omega($ $)$.

We reduced Problem B (the ____________________ problem) to an instance of problem A (the ____________________ problem)

Therefore, problem A (the ____________________ problem) is at least as hard as problem B (the ____________________ problem)

So problem A (the ____________________ problem) also has a lower bound of $\Omega($ $)$

What about Big-Θ?

Example 2: Euclidean Minimum Spanning Tree (MST)

Problem statement: given n points in the plane, construct a tree of minimum total length whose vertices are the given points.

A problem with a known lower bound to use to establish a lower bound for Euclidean MST:

Goal: reduce an instance of the ____________________ problem

to an instance of the ____________________ problem

Element uniqueness problem input: a set of numbers $x_1, x_2, \cdots, x_n$

Element uniqueness problem known lower bound: $\Omega(n \log n)$, tight

Steps:

1. Transform the element uniqueness inputs into a set of inputs for Euclidean MST (which is a set of points in the plane).
2. Solve Euclidean MST on that input
3. Use this solution to get a solution to element uniqueness

What did we show?