Reductions, Recursion and Divide and Conquer

Lecture 10

Tuesday, September 29, 2020

LaTeXed: September 1, 2020 12:55

10.1
Brief intro to the RAM model

Algorithms and Computing

1. Algorithm solves a specific problem.
2. Steps/instructions of an algorithm are simple/primitive and can be executed mechanically.
3. Algorithm has a finite description; same description for all instances of the problem
4. Algorithm implicitly may have state/memory

A computer is a device that

1. implements the primitive instructions
2. allows for an automated implementation of the entire algorithm by keeping track of state

Models of Computation vs Computers

1. Model of Computation: an idealized mathematical construct that describes the primitive instructions and other details
2. Computer: an actual physical device that implements a very specific model of computation

In this course: design algorithms in a high-level model of computation.

Question: What model of computation will we use to design algorithms?

The standard programming model that you are used to in programming languages such as Java/C++. We have already seen the Turing Machine model.

Models of Computation vs Computers

1. Model of Computation: an idealized mathematical construct that describes the primitive instructions and other details
2. Computer: an actual physical device that implements a very specific model of computation

In this course: design algorithms in a high-level model of computation.

Question: What model of computation will we use to design algorithms?

The standard programming model that you are used to in programming languages such as Java/C++. We have already seen the Turing Machine model.

Unit-Cost RAM Model

Informal description:

1. Basic data type is an integer number
2. Numbers in input fit in a word
3. Arithmetic/comparison operations on words take constant time
4. Arrays allow random access (constant time to access $A[i]$)
5. Pointer based data structures via storing addresses in a word

Example

Sorting: input is an array of n numbers

1. input size is n (ignore the bits in each number),
2. comparing two numbers takes $O(1)$ time,
3. random access to array elements,
4. addition of indices takes constant time,
5. basic arithmetic operations take constant time,
6. reading/writing one word from/to memory takes constant time.

We will usually not allow (or be careful about allowing):

1. bitwise operations (and, or, xor, shift, etc).
2. floor function.
3. limit word size (usually assume unbounded word size).

Caveats of RAM Model

Unit-Cost RAM model is applicable in wide variety of settings in practice. However it is not a proper model in several important situations so one has to be careful.

1. For some problems such as basic arithmetic computation, unit-cost model makes no sense. Examples: multiplication of two n-digit numbers, primality etc.
2. Input data is very large and does not satisfy the assumptions that individual numbers fit into a word or that total memory is bounded by 2^k where k is word length.
3. Assumptions valid only for certain type of algorithms that do not create large numbers from initial data. For example, exponentiation creates very big numbers from initial numbers.

Models used in class

In this course when we design algorithms:

1. Assume unit-cost RAM by default.
2. We will explicitly point out where unit-cost RAM is not applicable for the problem at hand.
3. Turing Machines (or some high-level version of it) will be the non-cheating model that we will fall back upon when tricky issues come up.

THE END

...

(for now)