CSC3310 Algorithms

Problems and Algorithms

William Retert

Milwaukee School of Engineering

Algorithms

Algorithm a step-by-step procedure for performing a task in a finite amount of time.

In this class:

- Common algorithms
- Algorithmic problem-solving techniques
- Analysis of algorithms

Multiplication

Let’s multiply two 4-digit numbers!

Multiplication

Let's multiply two 4-digit numbers!

- "Chalkboard" algorithm
- Lattice algorithm

Multiplication

Let's multiply two 4-digit numbers!

- "Chalkboard" algorithm
- Lattice algorithm
- Both multiply individual digits and add according to place

$$x = X[0, \ldots, n]$$

$$y = Y[0, \ldots, m]$$

$$x \times y = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} X[i] \times Y[j] \times 10^{i+j}$$

- $O(m \cdot n)$ single-digit multiplications

Peasant Multiplication

procedure PEASANTMULTIPLY(x,y)
    $res \leftarrow 0$
    while $x > 0$ **do**
        if x is odd **then**
            $res \leftarrow res + y$
        end if
        $x \leftarrow \lfloor x \div 2 \rfloor$
        $y \leftarrow y + y$
    end while
end procedure

- Different basic math operations!
  - double
  - halve

Peasant Multiplication

procedure PEASANTMULTIPLY(x,y)
    $res \leftarrow 0$
    while $x > 0$ **do**
        if x is odd **then**
            $res \leftarrow res + y$
        end if
        $x \leftarrow \lfloor x \div 2 \rfloor$
        $y \leftarrow y + y$
    end while
end procedure

- Different basic math operations!
- Correctness

$$x \times y = \begin{cases} \lfloor x \div 2 \rfloor \cdot (y + y) \text{if } x \text{ is even} \\ \lfloor x \div 2 \rfloor \cdot (y + y) + y \text{if } x \text{ is odd} \end{cases}$$

Peasant Multiplication

procedure PEASANTMULTIPLY(x,y)
    $res \leftarrow 0$
    while $x > 0$ **do**
        if x is odd **then**
            $res \leftarrow res + y$
        end if
        $x \leftarrow \lfloor x \div 2 \rfloor$
        $y \leftarrow y + y$
    end while
end procedure

- Different basic math operations!
- Loop runs $\lceil \lg x \rceil$ times (logarithm base 2)
- The work needed to add a number to itself depends on representation. If it is standard place representation, then it is proportional to $\log y$.
- Peasant multiplication is also $O(m \cdot n)$

Describing an Algorithm

Algorithm a step-by-step procedure for *solving a problem* in a finite amount of time.

When *describing* an algorithm, we want to provide:

- Description of the problem it solves
- The steps of the algorithm
- Proof that the algorithm is correct
- Efficiency of the algorithm

Describing an Algorithm

Algorithm a step-by-step procedure for *solving a problem* in a finite amount of time.

When *describing* an algorithm, we want to provide:

- **Description of the problem it solves**
  Target to a *user* of the algorithm. What do they need to know to *apply* the algorithm without knowing how it works?
  - Inputs, formally defined
  - Outputs, in terms of inputs
- The steps of the algorithm
- Proof that the algorithm is correct
- Efficiency of the algorithm

Describing an Algorithm

Algorithm a step-by-step procedure for *solving a problem* in a finite amount of time.

When *describing* an algorithm, we want to provide:

- Description of the problem it solves
- **The steps of the algorithm**
  - Specify the steps precisely
  - Use pseudocode
  - Use code constructs rather than (ambiguous!) English descriptions
  - But remember–your audience is **other people.** Write steps that you can explain.
- Proof that the algorithm is correct
- Efficiency of the algorithm

Describing an Algorithm

Algorithm a step-by-step procedure for *solving a problem* in a finite amount of time.

When *describing* an algorithm, we want to provide:

- Description of the problem it solves
- The steps of the algorithm
- Proof that the algorithm is correct
    - Loop invariants
    - induction
    - Often just a proof sketch
- Efficiency of the algorithm

Describing an Algorithm

Algorithm a step-by-step procedure for *solving a problem* in a finite amount of time.

When *describing* an algorithm, we want to provide:

- Description of the problem it solves
- The steps of the algorithm
- Proof that the algorithm is correct
- Efficiency of the algorithm
  - Also called "Time Complexity."
  - Generally written as a function of the size of the input (and possibly other factors)

$$T(n) = 3n^2 + \log n + 6$$

  - Number of primitive operations.
  - Summarized using asymptotic notation

Describing an Algorithm

Algorithm a step-by-step procedure for *solving a problem* in a finite amount of time.

When *describing* an algorithm, we want to provide:

- Description of the problem it solves
- The steps of the algorithm
- Proof that the algorithm is correct
- Efficiency of the algorithm

Target your explanations to a competent novice

- Work through the details. Expect your reader to take your words literally.
- Do not assume the audience shares your intuition. Things that are “obvious” to you still need to be written down and explained coherently.
- Similarly, provide evidence in your arguments for correctness and efficiency.

Example: Sorting

Name SORTING

Description Given a sequence of n values, return the same values in sorted order

Input A sequence of $A = \langle a_0, a_1, \ldots, a_{n-1} \rangle$

Output A permutation of A $\langle a'_0, a'_1, \ldots, a'_{n-1} \rangle$ such that $a'_0 \leq a'_1 \leq \cdots \leq a'_{n-1}$

Example: Find Minimum

Name FindMinimumValue

Description Given a sequence of n values, return a minimal value from the sequence

Input A sequence of $A = \langle a_0, a_1, \ldots, a_{n-1} \rangle$

Output A value a_i such that a_i is in the sequence, and $a_i \leq a_j$ for all $0 \leq j < n$

Example: Sorted Merge

Name SortedMerge

Description Given two *sorted* sequences with m and n values respectively, combine them into a single sorted sequence with $m + n$ values.

Input Two sequences $A = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ and $B = \langle b_0, b_1, \ldots, b_{m-1}$ such that $a_0 \leq a_1 \leq \cdots \leq a_{n-1}$ and $b_0 \leq b_1 \leq \cdots \leq b_{m-1}$

Output A permutation of AB $\langle c_0, c_1, \ldots, c_{m+n-1} \rangle$ such that $c_0 \leq c_1 \leq \cdots \leq c_{m+n-1}$

Computational Problems

Decision For a given input, answer “yes” or “no”
Given an array A and a value v, determine whether the value is stored in the array.

Search Compute some answer in relation to the input
Given an array A and a value v, find an index at which the value is stored.

Counting Determine *how many* answers relate to the input
Given an array A and a value v, determine how many times the value appears in the array.

Optimization Find a “best possible” solution
Given an array $A[0, \ldots, n-1]$, find the smallest value in A

Functional Produce an output for each valid input
Given an array $A[0, \ldots, n-1]$, return an array with the same elements in *sorted* order

Computational Problems

Decision For a given input, answer “yes” or “no”
Given an array A and a value v, determine whether the value is stored in the array.

Search Compute some answer in relation to the input
Given an array A and a value v, find an index at which the value is stored.

Counting Determine *how many* answers relate to the input
Given an array A and a value v, determine how many times the value appears in the array.

Optimization Find a “best possible” solution
Given an array $A[0, \ldots, n-1]$, find the smallest value in A

Functional Produce an output for each valid input
Given an array $A[0, \ldots, n-1]$, return an array with the same elements in *sorted* order

Computational Problems

Decision For a given input, answer “yes” or “no”
Given an array A and a value v, determine whether the value is stored in the array.

Search Compute some answer in relation to the input
Given an array A and a value v, find an index at which the value is stored.

Counting Determine *how many* answers relate to the input
Given an array A and a value v, determine how many times the value appears in the array.

Optimization Find a “best possible” solution
Given an array $A[0, \ldots, n-1]$, find the smallest value in A

Functional Produce an output for each valid input
Given an array $A[0, \ldots, n-1]$, return an array with the same elements in *sorted* order

Computational Problems

Decision For a given input, answer "yes" or "no"
Given an array A and a value v, determine whether the value is stored in the array.

Search Compute some answer in relation to the input
Given an array A and a value v, find an index at which the value is stored.

Counting Determine *how many* answers relate to the input
Given an array A and a value v, determine how many times the value appears in the array.

Optimization Find a "best possible" solution
Given an array $A[0, \ldots, n-1]$, find the smallest value in A

Functional Produce an output for each valid input
Given an array $A[0, \ldots, n-1]$, return an array with the same elements in *sorted* order

Computational Problems

Decision For a given input, answer "yes" or "no"
Given an array A and a value v, determine whether the value is stored in the array.

Search Compute some answer in relation to the input
Given an array A and a value v, find an index at which the value is stored.

Counting Determine *how many* answers relate to the input
Given an array A and a value v, determine how many times the value appears in the array.

Optimization Find a "best possible" solution
Given an array $A[0, \ldots, n-1]$, find the smallest value in A

Functional Produce an output for each valid input
Given an array $A[0, \ldots, n-1]$, return an array with the same elements in *sorted* order