CS140: Algorithms

Z Sweedyk
Lecture 0
1/16/01

Today

- Who Am I?
- Who Are You?
- Course Overview
- Some Basics

Course Overview

1. How to analyze algorithms
2. How to design algorithms
3. NP-completeness

Course Requirements

- Homework
- Exams
- Class participation

Homework

- Assignments will be posted to the web page on Tuesday and Thursday
- Assignments are due at the start of the next class period.
- Solutions will be posted at due time so no late homework will be accepted
- Solutions should be prepared in LaTex

Exam Due Dates
(Tentative Schedule)

- Exam I: Feb. 13
- Exam II: Mar. 27
- Exam III: Apr 24
- Final: May 12

- Exams will be take-home, timed, closed book

Course Requirements – class participation

- Show up to class
- Speak up in class
- Hand in daily “worksheets”

Advice

- Stay on top of things
- Seek help when necessary

The Problem

Computational Problem: Specified by input/output pair

Sorting

- Sorting Integers in Ascending Order (SIAO):
  Input: A list of integers
  Output: The input integers sorted in ascending order

- Example:
  Input: 5,3,8,1,2
  Output:

The Algorithm

- Computational Problem: Specified by input/output pair.

- Algorithm: Well-defined sequence of computational steps that produce a correct output for every valid input.

What should an algorithm for SIAO do?

- Example 1:
  - Input: 5,3,8,1,2
  - Output:

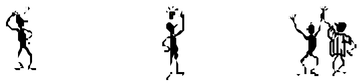
- Example 2:
  - Input: 3,a,5.27,mudder
  - Output:

Software Development

The problem: Huh?

The idea: A-ha!

The program: Ta-da!



The algorithm exists somewhere between a-ha and ta-da.

An Algorithm for SIAO?

```
Sort1(S)
    While there are integers x and y in S such that x precedes y
      in S and x > y
        Swap x and y in S
    Return S
```


Sort1 Example

Input: 5, 3, 8, 1, 2

Swap 3 and 2: 5, 2, 8, 1, 3

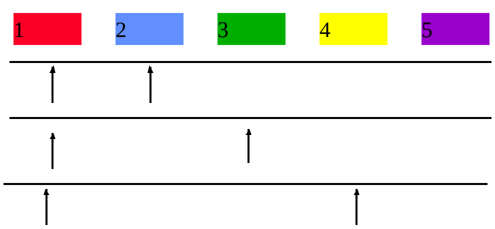
Swap 5 and 3: 3, 2, 8, 1, 5

Is Sort1 an algorithm?

Is Sort1(S) well-defined?

No! We need to specify a selection rule.

Order on pairs of indices



Etc.

An Algorithm for SIAO?

```
Sort2(S)
    Assume a fixed order on pairs of elements in S
    While there are integers x and y in S such that x precedes y
      in S and x >y
        Choose first pair x,y that is out of order
        Swap x and y in S
    Return S
```

Sort2 Example

Input: 5, 3, 8, 1, 2

Swap 5 and 3: 3,5,8,1,2
Swap 3 and 1: 1,5,8,3,2

Is Sort2 an algorithm?

Is Sort2(S) well-defined? Yes.

Does it produce the correct output for any valid input?

Proof of correctness

- When the algorithm halts S is sorted.
- The algorithm halts on all input.
  - How can we measure the progress the algorithm makes from a swap?

Claim

- The number of "out-of-order" pairs decreases with each swap.
- Progress: Eventually we'll have 0 "out-of-order" pairs and the algorithm will halt.

Illustration of claim

Input:	5, 3, 8, 1, 2	#out-of-order pair: 7
Swap 5 and 3:	3, 5, 8, 1, 2	6
Swap 3 and 1:	1, 5, 8, 3, 2	5

Proof of claim:

- Suppose the algorithm chooses to swap x and y:
  $u_0, \ldots, u_i, x, v_0, \ldots, v_j, y, w_0, \ldots, w_k$
- Change in status of a pair:
  - Out-of-order → Ordered
  - Ordered → Out-of-order
- Pairs to consider
  - u-u, u-x, u-v, u-y, u-w,
  - x-v, x-y, x-w,
  - v-v, v-y, v-w
  - y-w, w-w

Ordered→unordered

- Suppose an x -v pair goes from ordered to out-of-order.
- Then ______________________________

Ordered→unordered

- Suppose an x -y pair goes from ordered to out-of-order.
- Then _____________________________

Ordered→unordered

- Suppose an v -y pair goes from ordered to out-of-order.
- Then ______________________________

Proof

- The x-y pair goes from out-of-order to ordered.
- For every pair that goes from ordered to out-of-ordered _______________________.

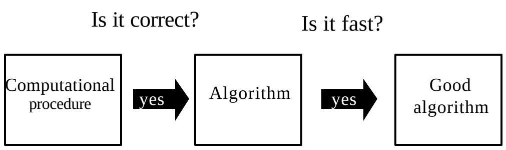
Is Sort2 a good algorithm?

- Is it easy to understand?
- Is it easy to implement?
- Is it fast?
- Is it space-efficient?

How fast is Sort2?

- How many swaps can the algorithm make?
- How many comparisons does the algorithm need to make to find a pair to swap?
- How many comparisons does the algorithm make?
- The running time is ___________

CS140: Two questions



Another algorithm for SAIO?

```
Sort3(S)
    If ||S||≤1
        Return:   S
    Else
        Return:   Sort3(S\max-element(S)),max-element(S)
```


Proof of correctness

(Prove Def. 1 ≈ Def. 2)

– Def. 1 (non-recursive)

$s_0, s_1, \ldots, s_n$ is sorted if for every i,j such that $i<j$ it holds that $s_i \leq s_j$

– Def. 2 (recursive)

The list $s_0, s_1, \ldots, s_n$ is sorted if

1. n=0, or
2. n>0 and $s_0, s_1, \ldots, s_{n-1}$ is sorted and for all $i<n$ it holds that $s_n \geq s_i$

Example: Sort3(3,1,5,2,4)

Sort3(3,1,5,2,4) = Sort3(3,1,2,4),5
= Sort3(3,1,2),4,5
= Sort3(1,2),3,4,5
= Sort3(1),2,3,4,5
= 1,2,3,4,5

Recursive Algorithms

What about Sort3?

```
Sort3(S)
    If ||S||≤1
        Return:   S
    Else
        Return:   Sort3(S\max-element(S)),max-element(S)
```

Let T(n) be the running time of Sort3:

$T(1) = c_2$
$T(n) = c_1 n + T(n-1)$, n>1

Unwinding

$T(n) = c_1 n + T(n-1)$
$= c_1 n + c_1(n-1) + T(n-2)$
$= c_1 n + c_1(n-1) + c_1(n-2) + T(n-3)$

⋮

Basic skills

- LaTex – HW0
- Ch 2.2 of CLR