Motivations of Data Structures and Algorithms

Hsuan-Tien Lin

Dept. of CSIE, NTU

March 2, 2021

What We Have Done

- algorithm $\sim$ programming recipe
- algorithm: input (problem/data), output (correctness), definiteness (instruction), finiteness (efficiency), effectiveness (computability)
- sequential search algorithm for getMinIndex: loop through array to find the minimum one

Correctness Proof of Algorithm

C Version

```
/*return index to
  min. element
  in arr[0]...arr[len-1]*/
int getMinIndex
    (int arr[], int len){
  int i;
  int m=0;
  for(i=0;i<len;i++){
    if (arr[m] > arr[i]){
      m = i;
    }
  }
  return m;
}
```

Theorem

`getMinIndex` returns `m` such that `arr[m]` is the smallest among `arr[0]`, `arr[1]`, ..., `arr[len-1]`.

$\forall i \in \{1, 2, \ldots, len-1\}, arr[m] \le arr[i]$

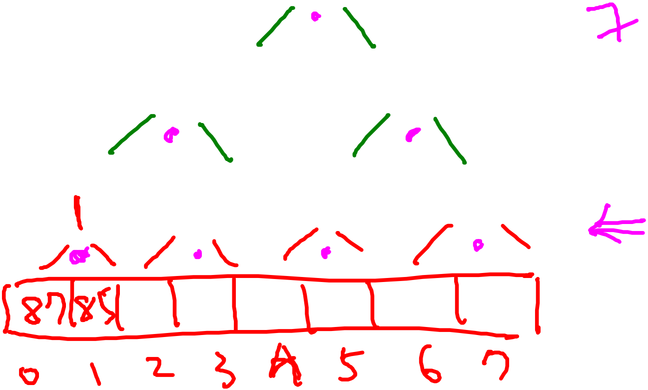
Lemma (loop invariance)

*After the `i`-th iteration of the **`for`** loop, `arr[m]` is always the smallest among `arr[0]`, `arr[1]`, ..., `arr[i]`.*

- *true when `i=0`?*
- *true when `i=k` $\Longrightarrow$ true when `i=k+1`?*

e.g. mathematical induction proves the loop invariance lemma (and hence theorem)—discrete math helps!

Efficiency of Algorithm



knockout tournament for `getMinIndex`: not much faster overall, but possibly faster if done in parallel

Expressing Algorithms with Pseudo Code

Pseudo Code for `getMinIndex`

C Version

```c
/* return index to min. element
   in arr[0] ... arr[len-1] */
int getMinIndex
    (int arr[], int len){
  int i;
  int m=0;
  for(i=0;i<len;i++){
    if (arr[m] > arr[i]){
      m = i;
    }
  }
  return m;
}
```

Pseudo Code Version

GET-MIN-INDEX(A)

```
m = 1
for i = 2 to A.length
        // update if i-th element smaller
        if A[m] > A[i]
                m = i
return m
```

pseudo code: “spoken language” of programming

Bad Pseudo Code: Too Detailed

Unnecessarily Detailed

```
GET-MIN-INDEX(A)
m = 1
for i = 2 to A.length
        // update if i-th element smaller
        Am = A[m]
        Ai = A[i]
        if Am > Ai
                m = i
        else
                m = m
return m
```

Concise

```
GET-MIN-INDEX(A)
m = 1
for i = 2 to A.length
        // update if i-th element smaller
        if A[m] > A[i]
                m = i
return m
```

goal of pseudo code: communicate efficiently

Bad Pseudo Code: Too Mysterious

Unnecessarily Mysterious

```
GET-MIN-INDEX(A)
x = 1
for xx = 2 to A.length
      // update if xx-th element smaller
      if A[x] > A[xx]
            xx = x
return aa
```

Clear

```
GET-MIN-INDEX(A)
m = 1 // store current min. index
for i = 2 to A.length
      // update if i-th element smaller
      if A[m] > A[i]
            m = i
return m
```

goal of pseudo code: communicate correctly

Bad Pseudo Code: Too Abstract

Unnecessarily Abstract

```
GET-MIN-INDEX(A)
  run a loop through A
  that updates m in every iteration
  return m
```

Concrete

```
GET-MIN-INDEX(A)
  m = 1 // store current min. index
  for i = 2 to A.length
      // update if i-th element smaller
      if A[m] > A[i]
          m = i
  return m
```

goal of pseudo code: communicate effectively

Good Pseudo Code of `Selection-Sort`

```
SELECTION-SORT(A)
for i = 1 to A.length
        s = GET-MIN-INDEX(A[i . . end])
        SWAP(A[i], A[s])
return A, which has been sorted in place
```

no "formal definition" and depends on the speaker/listener
(follow textbook if you really need a "definition")

Introduction of Data Structures

What is Data Structure?

scheme of organizing data
within computer

How to Organize 400 Exam Sheets?

different use cases

$\Longrightarrow$ different organization scheme (data structure)

Good Algorithm Needs Proper Data Structure

if having data structure such that `getMinIndex` *faster*,
$\Longrightarrow$ `SelSort` *also faster* (we will see)

algorithm :: data structure $\sim$ recipe :: kitchen structure

Good Data Structure Needs Proper Accessing Algorithms: `get`, `insert`

rule of thumb for speed: often-`get` $\Leftrightarrow$ "nearby"

Good Data Structure Needs Proper Maintenance

Algorithms: `construct`, `update`, `remove`

hidden “cost” of data structure: maintenance effort

Why Data Structures and Algorithms?

Why Data Structures and Algorithms?

network

... Storage

bandwidth

computation

use storage/computation resources properly $\Longrightarrow$ good program

Proper Use: Tradeoff of Different Factors

understand tradeoff $\Longrightarrow$ good program

Different Tradeoff on Different Platforms

important to learn other CS subjects

Programming $\neq$ Coding

programming :: building house $\sim$ coding :: construction work

C Programming versus DSA

test
modualize
tradeoff (theory/practice)

moving from coding to designing