Zubair Abid
20171076

POPL W13 Friday Quiz

2020 - 11 - 09

Resolving Ambiguities

A1: 1. Looking for consecutive subsequences happens linearly, there is no oracle picking it.

A2: 1. When picking "adjacent subsequences" we do not pick overlapping sub sequences

1. TS1:

States: Array (A), containing elements with value = Comparable Data
sequence = { false, 1, 2 }

Input: Unsorted Array A_0

Output: Sorted Array A_F

∃ function "compare" to compare two values in Array, and returns the value that fits

Looking at the various machines to sort:

1. Sorted sequence identifier.

Input: $A_x[i:]$ s.t. $0 \le i < |A_x|$
and $A_x[i...]$. sequence = false

$$sub1 = \{ j \mid compare(A_x[j-1], A_x[j]) = A_x[j-1] \text{ AND } A_x[j-1] \in sub1 \}$$

$$sub2 = \{ j \mid compare(A_x[j-1], A_x[j]) = A_x[j-1] \text{ AND } (A_x[j-1] \in sub2 \text{ OR } (A_x[j-1] \in sub1 \text{ AND } A_x[j] \notin sub1)) \}$$

$$\boxed{A_{x+1} = A_x[i + sub1].\,sequence = 1 \text{ AND } A_x[i + sub2].\,sequence = 2}$$

2. Subsequence Merger

Input: A_x s.t. $\exists\, k \in A_x$ and k. sequence = 1
and $\exists\, l \in A_x$ and l. sequence = 2

$$sub1 = \{ A[j] \mid A[j].\,sequence = 1 \}$$

$$sub2 = \{ A[j] \mid A[j].\,sequence = 2 \}$$

$\text{new} = \{ k \mid k = \text{compare}(\text{sub1}[\ldots], \text{sus2}[\ldots]) \}$

$\boxed{A_{x+1} = A_x.\text{replace}(\text{new})}$

TS2:

1. <u>Subsequence picker</u>

Input: $A_x[i:]$ s.t. $0 \leq i < |A_x|$ and $A_x[i\ldots].\text{sequence} = \text{false}$.

$n = \{1, 2, 4, \ldots n\}$

$\text{sub1} = A_x[i : i+n]$

$\text{sub2} = A_x[i+n+1 ; i+2n]$

$\boxed{A_{x+1} = A_x[i + \text{sub1}].\text{sequence} = 1 \text{ AND } A_x[i + \text{sub2}].\text{sequence} = 2}$

2. <u>Subsequence Merger</u>

Input: A_x s.t. $\exists\, k \in A_x$ and $k.\text{sequence} = 1$ and $\exists\, l \in A_x$ and $l.\text{sequence} = 2$

$\text{sub1} = \{ A[j] \mid A[j].\text{sequence} = 1 \}$

$\text{sub2} = \{ A[j] \mid A[j].\text{sequence} = 2 \}$

$\text{new} = \{ k \mid k = \text{compare}(\text{sub1}[\ldots], \text{sus2}[\ldots]) \}$

$\boxed{A_{x+1} = A_x.\text{replace}(\text{new})}$

2. A1 and A2 are both more similar to mergesort than either is to bubble sort.

Do they bear similarities? Yes. Both are similar to merge sort: the key component of both is merging already-sorted sequences. A2 is, in fact, merge sort — with the lack of the constraint that skips over testing for already sorted sequences. A1 is more alike in the sense of picking sorted sequences, although the minimum unit for sorting is not, here, a one-length subarray.

The similarity to bubblesort is, in my observation, negligible. The only one being that instead of a formal "divide and conquer" approach these algorithms work by iterative looping.

TS1 and TS 2 both share their merger machines with merge sort. They don't really share any machines with the bubble sort machines.

3. Bubblesort:

Bubblesort is three simple machines:

1. Compare and Swap
2. Update index
3. Update iteration

1. Compare and Swap
   - takes index i, Compares A[i] to A[i+1] if possible, and swaps them if the comparison fails.
2. Update Index:
   - Updates i to i+1. If the update falls outside bounds, it calls Update iteration.
3. Update iteration:
   - Updates i to 0. Reduces the limits by 1.

Merge sort:

Mergesort has two primary machines

1. Splitting the array in 2 parts
2. Merging sorted subsequences.

1 Splitting the array in 2 parts

- Takes the array, takes the first half and call the algorithm on the subarray again.

2. Merging sorted subsequences

- Takes two sorted sequences and merges them.

a) Key components of each:

The key component for bubblesort is the compare and swap machine, and with mergesort it is the merge machine and the split machine.

Similarities: They both contain the same input and final output.

Differences: Everything else.

Primarily, the merge vs the compare/swap system.