

Chapter 10: What's The Plan?: Algorithmic Thinking

Fluency with Information Technology
Third Edition

by
Lawrence Snyder



Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Algorithm

- A precise, systematic method for producing a specified result
- We have already seen several:
 - Recognition of a button click (Ch.1)
 - Placeholder technique (Ch.2)
 - Hex to bits (Ch.8)

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

10-2

Algorithms in Everyday Life

- Some algorithms are learned—arithmetic
- Some we figure out ourselves—looking up a phone number
- Others require written instructions—recipe, assembly instructions, driving directions

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

10-3

Five Essential Properties of Algorithms

1. Input specified
 - Data to be transformed during the computation to produce the output
 - Must specify type, amount, and form of data
2. Output specified
 - Data resulting from the computation—intended result
 - It is possible to have no output

Copyright © 2008 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

10-4

Five Essential Properties (cont'd)

3. Definiteness

- Specify the sequence of events
- Details of each step, including how to handle errors

4. Effectiveness

- The operations are doable

5. Finiteness

- Must eventually stop

Language in Algorithms

• *Natural language*

- For people, we use a natural language like English
- Ambiguity is common in natural language

• *Programming Language*

- Formal languages designed to express algorithms
- Precisely defined; no ambiguity

The Context of a Program

- Program can fulfill five properties of an algorithm, be unambiguous, and still not work right because it is executed in the wrong context
 - e.g., last name in Western countries means family name; in Asian countries it may mean given name
- Context matters: Driving instructions
 - "From the Limmat River go to Bahnhof Strasse and turn right."
 - Assumes you are traveling in a specific direction. If you are not, the directions will fail.

Program vs. Algorithm

- A program is an algorithm that has been customized to solve a specific task under a specific set of circumstances using a specific language
- Algorithm is a general method; program is a specific method

An Algorithm: *Alphabetize CDs*

- Imagine CDs in a slotted rack, not organized
- You want to alphabetize by name of group, performing musician, or composer
- How do you solve this problem?

Alphabetize CDs (cont'd)

- Input: An unordered sequence of CDs filling a slotted rack
- Output: The same CDs in the rack in alphabetical order
- Instructions:
 1. Use the term Artist_Of to refer to the name or the group or musician or composer on a CD
 2. Decide which end is the beginning of the sequence and call the slot at that end the Alpha slot
 3. Call the slot next to Alpha the Beta slot

Alphabetize CDs (cont'd)

4. If the Artist_Of the CD in Alpha comes later in alphabet than the Artist_Of in Beta, swap the CDs; otherwise continue on
5. If there is a slot following Beta, begin calling it Beta and go to Instruction 4; otherwise continue on
6. If there are two or more slots following Alpha, begin calling the slot following Alpha, Alpha and begin calling the slot following it Beta, and go to Instruction 4; otherwise stop

Instruction 1

- Use the term Artist_Of to refer to the name of the group or musician or composer on a CD
 - This gives a name to the operation of locating the name used for alphabetizing

Instruction 2

- Decide which end of the rack is the beginning of the alphabetic sequence and call the slot at that end the Alpha slot
 - Gives the process a starting point. Defines the word Alpha. At the start, Alpha refers to the first slot in the sequence; as the algorithm progresses it refers to successive slots

Instruction 3

- Call the slot next to the Alpha slot the Beta slot
 - Gives Beta its initial meaning. Alpha and Beta have no inherent meaning; they are chosen by the programmer to name slots in the rack

Instruction 4

- If the Artist_Of the CD in the Alpha slot comes later in the alphabet than the Artist_Of the CD in the Beta slot, swap the CD's; otherwise continue on
 - This is the workhorse instruction of the algorithm. It compares the names of the artists of the CDs in Alpha and Beta and, if necessary, swaps them so that they are in the correct order

Instruction 5

- If there is a slot following the Beta slot, begin calling it the Beta slot and go to Instruction 4; otherwise, continue
 - Re-defines Beta slot so that it refers to the next slot in the sequence, if any. Then Instruction 4 can be executed again, comparing a different pair of CDs

Instruction 6

- If there are two or more slots following the Alpha slot, begin calling the slot following the Alpha slot Alpha and the slot following it Beta, and go to Instruction 4; otherwise stop
 - By the time we get to this instruction, the alphabetically earliest CD is in the Alpha slot. We advance Alpha to the next slot and sweep through the CD rack again, locating the alphabetically earliest CD. When there are no longer enough slots to call one Alpha and one Beta, the entire rack has been alphabetized and the algorithm stops

Analyzing Alphabetize CDs Algorithm

- Illustrates the five basic properties of algorithms
 - Inputs and Outputs were listed
 - Each instruction was defined precisely (definiteness)
 - Operations are effective because they are simple and mechanically doable
 - Alphabetizing is mechanical, so our algorithm is effective
 - Finiteness is satisfied because there are only a finite number of slots that can be paired, so instructions 4, 5, and 6 cannot be repeated indefinitely

A Deeper Analysis

- Structural features
 - Two instructions, 5 and 6, in which the agent is directed to go back and repeat instructions. This is called a *loop*
 - Loops and Tests
 - A loop must include a test to determine whether the instructions should be repeated one more time
 - Assumptions
 - We assume that
 - The CD rack is full (instructions do not handle the case of an empty slot)
 - The word "following" means a slot further from the start point

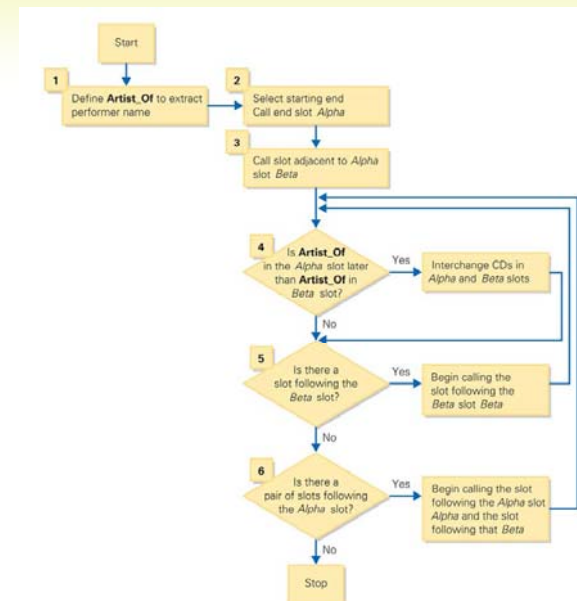


Figure 10.3. Flowchart of Alphabetize CDs. Operations are shown in rectangles; decisions are shown in diamonds. Arrows indicate the sequencing of the operations.

The Exchange Sort Algorithm

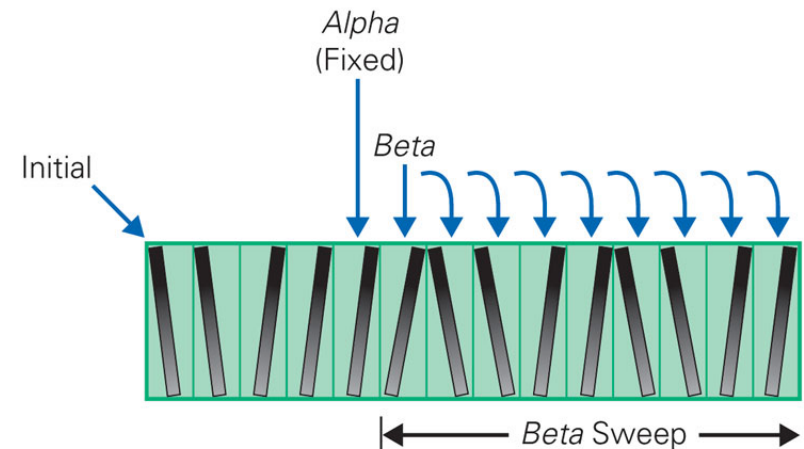
- The Alphabetize CDs example illustrates the standard Exchange Sort algorithm
 - The idea of comparing pairs of items chosen in a particular way, exchanging them if they are out of order, and continuing to sweep through the items
 - We could use the same algorithm to sort on a different principle

Abstraction in Algorithmic Thinking

- Thinking of parts of the algorithm's behavior as whole units rather than individual instructions
- Work proportional to n^2
 - Where n is the number of items to sort
- Why?

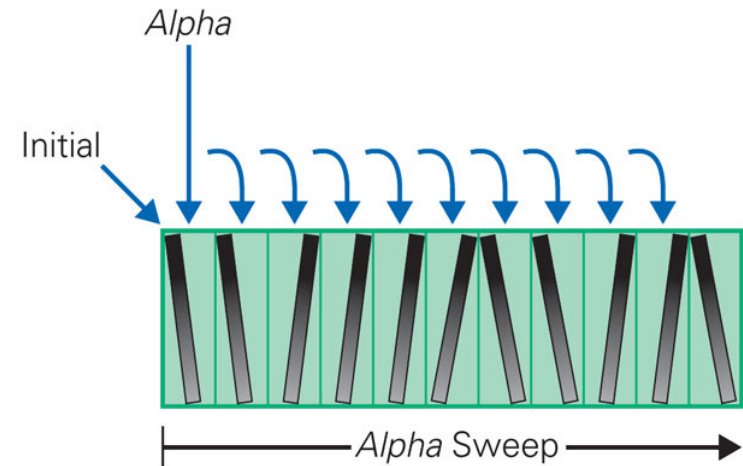
Beta Sweep Abstraction

- Instructions 4 and 5 illustrate abstract computational unit, which we'll call Beta Sweep
 - Operation of considering in order all CDs following a specific Alpha ($\sim \frac{1}{2}n$ comparisons on average)
 - While Alpha points to a fixed slot, Beta visits each slot following Alpha, in sequence, comparing its CD with the CD in the Alpha slot, and swapping when necessary
 - Has the effect of finding the next CD in order and moving it to the Alpha slot



Alpha Sweep Abstraction

- Alpha sweeps from the slot where alphabetization begins through all slots (except the last), performing the Beta sweep instructions each time
 - ~ n Alpha sweep operations



Properties of Alpha Sweep Abstraction

1. Exhaustive. Considers all CDs from the first to (but not including) the last
2. Nonredundant. No slot is assigned to Alpha more than once, so the process stops if the Beta sweep stops
3. Progressive. At the end of each Beta sweep, the alphabetically next earliest CD is in Alpha
4. Complete. When the last Beta sweep is completed, the CD in the last slot is later than the CD in the next-to-last slot
5. Goal-achieving. The alphabetically earliest CD is in the first slot at the end of the first Beta sweep, by its property 4 and the fact that all CDs are considered; thereafter, in every new position for Alpha the Beta sweep assigns the next earliest CD. The program alphabetizes.

Abstracting for Other Algorithms and Programs

- The Alpha sweep and Beta sweep abstractions are specific to the Exchange Sort algorithm and to programs derived from it
- Other algorithms and programs exhibit different behaviors and require different abstractions, which will have different properties
- Every situation is different, but the approach—abstracting the behavior and understanding the properties—is always the same

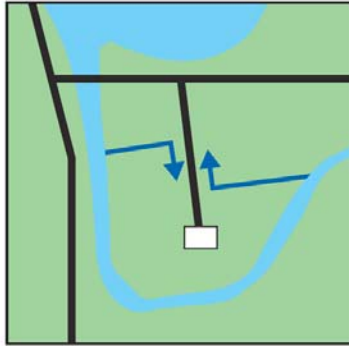
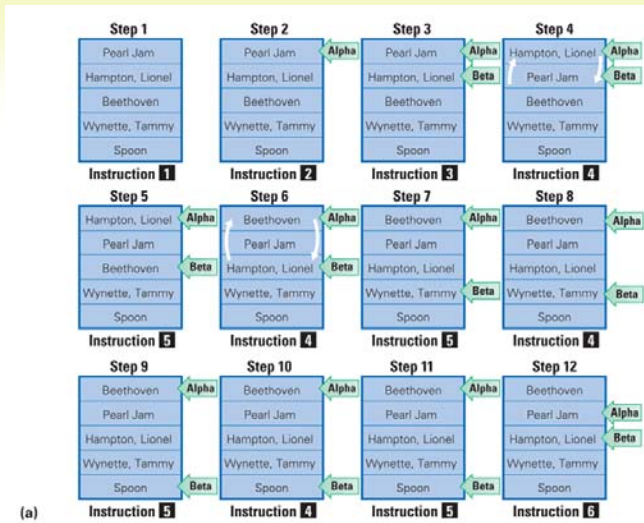


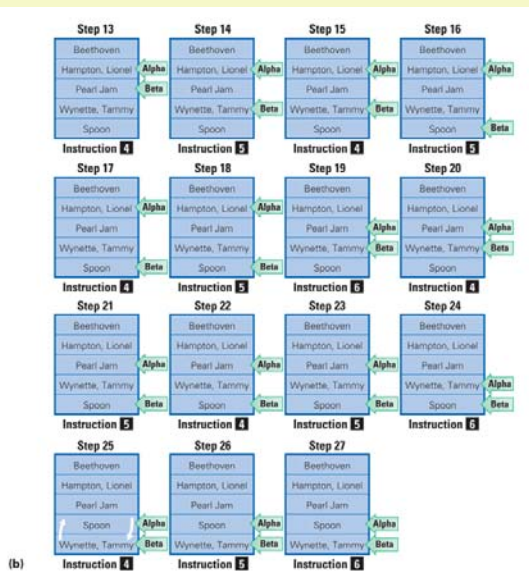
Figure 10.1. Diagram of approaching a street (Bahnhof Strasse) from different directions, giving the “turn right” instruction different meanings.



(a)

Figure 10.2. The steps of Alphabetize CDs algorithm. A snapshot of the CD rack is shown at the completion of each instruction. Notice how Beta sweeps through all of the slots following Alpha. After the first 11 steps, the alphabetically earliest CD, Beethoven, is in the Alpha slot.

(continues next page).



(b)

Figure 10.2. The steps of Alphabetize CDs algorithm. A snapshot of the CD rack is shown at the completion of each instruction. Notice how Beta sweeps through all of the slots following Alpha. After the first 11 steps, the alphabetically earliest CD, Beethoven, is in the Alpha slot. (continued).