1.8 Problem solving

Programming languages vs. problem solving

A chef may write a new recipe in English, but creating a new recipe involves more than just knowing English. Similarly, creating a new program involves more than just knowing a programming language. Programming is largely about ***problem solving***: creating a methodical solution to a given task.

The following are real-life problem-solving situations encountered by one of this material's authors.

Example 1.8.1: Solving a (nonprogramming) problem: Matching socks.

A person stated a dislike for matching socks after doing laundry, indicating there were three kinds of socks. A friend suggested just putting the socks in a drawer and finding a matching pair each morning. The person said that finding a matching pair could take forever: Pulling out a first sock and then pulling out a second, placing them back and repeating until the second sock matches the first could go on many times (5, 10, or more).

The friend provided a better solution approach: Pull out a first sock, then pull out a second, and repeat (without placing back) until a pair matches. In the worst case, if three kinds of socks exist, then the fourth sock will match one of the first three.

Feedback?

PARTICIPATION ACTIVITY 1.8.1: Matching socks solution approach.

Exactly three sock types A, B, and C exist in a drawer.

1) If sock type A is pulled first, sock type B second, and sock type C third, the fourth sock type must match one of A, B, or C.
   - (•) True
   - ( ) False

Correct

The sock type will be either A, B, or C, which matches one already pulled.

2) If socks are pulled one at a time and kept until a match is found, at least four pulls are necessary.
   - ( ) True
   - (•) False

Correct

If the first pull is A and the second is A, a match would be found in just 2 pulls, which is fewer than 4 pulls.

3) If socks are pulled two at a time and put back if not matching, and the process is repeated until the two pulled socks match, the maximum number of pulls is 4.
   - ( ) True
   - (•) False

Correct

No limit on the number of pulls exists. Of course, the odds of pulling a matching pair are 1/3, so a person might expect to find a match after 3 pulls, but 3 pulls is not guaranteed.

Feedback?

Example: Greeting people

PARTICIPATION ACTIVITY 1.8.2: Greeting people problem.

An organizer of a 64-person meeting wants to start by having every person individually greet each other person for 30 seconds. Indicate whether the proposed solution achieves the goal without using excessive time. Before answering, think of a possible solution approach for this seemingly simple problem.

1) Form an inner circle of 32 and an outer circle of 32, with people matched up. Every 30 seconds, have

Correct

Each person would meet the 32 people in the other circle, but not the other 31 in that person's own circle.

the outer circle shift left one position.

- ○ Yes
- ◉ No

2) Pair everyone randomly. Every 30 seconds, tell everyone to find someone new to greet. Do this 63 times.

- ○ Yes
- ◉ No

Correct

The approach uses excessive time: Finding a new person to greet will involve much wandering around.

3) Have everyone form a line. Then have everyone greet the person behind them.

- ○ Yes
- ◉ No

Correct

The person behind them is facing the wrong way, so not a single greeting occurs. (This is a common joke task that an organizer assigns to attendees).

4) Have everyone form a line. Have the first person greet the other 63 people for 30 seconds each. Then have the second person greet each other person for 30 seconds each (skipping anyone already met). And so on.

- ○ Yes
- ◉ No

Correct

Uses excessive time: 63 * 30 sec for the first person, then 62 * 30 sec for the second person, etc. Most people are just sitting around.

5) Form two lines of 32 each, with attendees matched up. Every 30 seconds, have one line shift left one position (with the person on the left end wrapping to right). Once the person that started on the left is back on the left, then

Correct

With two lines, 32 * 30 seconds is used. Then the lines split, yielding 4 lines (16 and 16, and another 16 and 16). 16 * 30 seconds is used. Then 8 lines (8 and 8, 8 and 8, 8 and 8, 8 and 8), so 8 * 30 sec. And so on. Total time is about (32 + 16 + 8 + 4 + 2) * 30 sec, or 62 * 30 sec. This solution is an example of a "recursive" solution (discussed elsewhere).

have each line split into two matched lines, and repeat until each line has just 1 person.

- (●) Yes
- ( ) No

Feedback?

Example: Sorting name tags

Example 1.8.2: Example: Sorting name tags.

1,000 name tags were printed and sorted by first name into a stack. A person wishes to instead sort the tags by last name. Two approaches to solving the problem are:

- Solution approach 1: For each tag, insert that tag into the proper location in a new last-name-sorted stack.
- Solution approach 2: For each tag, place the tag into one of 26 substacks, one for last names starting with A, one for B, etc. Then, for each substack's tags (like the A stack), insert that tag into the proper location of a last-name-sorted stack for that letter. Finally combine the stacks in order (A's stack on top, then B's stack, etc.).

Solution approach 1 will be very hard; finding the correct insertion location in the new sorted stack will take time once that stack has about 100 or more items. Solution approach 2 is faster, because initially dividing into the 26 stacks is easy, and then each stack is relatively small, so insertions are easier to do.

In fact, sorting is a common problem in programming, and the above sorting approach is similar to a well-known sorting approach called radix sort.

Feedback?

PARTICIPATION ACTIVITY 1.8.3: Sorting name tags.

1,000 name tags are to be sorted by last name by first placing tags into 26 unsorted substacks (for A's, B's, etc.), then sorting each substack.

1) If last names are equally distributed among the alphabet, what is the largest number of name tags in any one substack?

- 1
- (•) 39
- 1,000

Correct

1,000 / 26 is 39 (rounded up).

2) Suppose the time to place an item into one of the 26 sub-stacks is 1 second. How many seconds are required to place all 1000 name tags onto a sub-stack?

- 26 sec
- (•) 1,000 sec
- 26,000 sec

Correct

26 is not a large number; all substacks can be on a table and in order so a person can quickly (about 1 sec) toss each name tag onto the right substack.

3) When sorting each substack, suppose the time to insert a name tag into the appropriate location of a sorted N-item sub-stack is N * 0.1 sec. If the largest substack is 50 tags, what is the longest time to insert a tag?

- (•) 5 sec
- 50 sec

Correct

50 * 0.1 sec is 5 sec.

4) Suppose the time to insert a name tag into an N-item stack is N * 0.1 sec. How many seconds are required to insert a name tag into the appropriate location of a 500-item stack?

- 5 sec
- (•)

Correct

500 * 0.1 sec is 50 sec. The larger a stack, the harder is the stack to work with. Once the stack is large, each item will take many minutes to insert, leading to many hours of total work.

50 sec

Feedback?

A programmer usually should carefully create a solution approach *before* writing a program. Like English being used to describe a recipe, the programming language is just a description of a solution approach to a problem; creating a good solution should be done first.