PSEUDOCODE

CSC100 Introduction to programming in C/C++ (Spring 2023)

Marcus Birkenkrahe

April 14, 2023

Contents

1 README

- In this section of the course, we go beyond simple statements and turn to program flow and evaluation of logical conditions
- This section follows chapter 3 in Davenport/Vine (2015) and chapters 4 and 5 in King (2008)
- Practice workbooks, input files and PDF solution files in GitHub

2 Overview

- **Pseudocode** is a method to write down/analyze an *algorithm* or a *heuristic* without having to bother with *syntax* (like `&i` vs. `i`)
- The prefix pseudo- comes from Ancient Greek , meaning "lying", "false" or "untrue", as in "pseudoscience" or "pseudonym"
- Pseudocode does not need to compile or run so it is closer to a heuristic than to an exact algorithm.
- Code however needs to be exact and is always algorithmic.
- The answer is: always start with pseudocode before coding, and when you're stuck (not because of syntax ignorance) go back to pseudocode

3 Algorithms vs. heuristic

A flow diagram example that also illustrates when one might want to use an exact algorithm vs. a heuristic.

4 Example

- Example: player problem statement in **??** below:

  > "Drink a health potion when a character's health is 100 or less. If health reaches 100 or more, resume battle."

- Given the problem **??**, this is the pseudocode **??**[1]:

```
if health is less than 100
  Drink health potion
else
  Resume battle
end if
```

- The code in **??** would not compile as a C program (you can test yourself: which mistakes would the compiler find?[2])

- The conceptual "trick" with generating pseudocode from a prose description is to identify the **logical condition** so that you can perform a comparison (= apply a **conditional operator**)

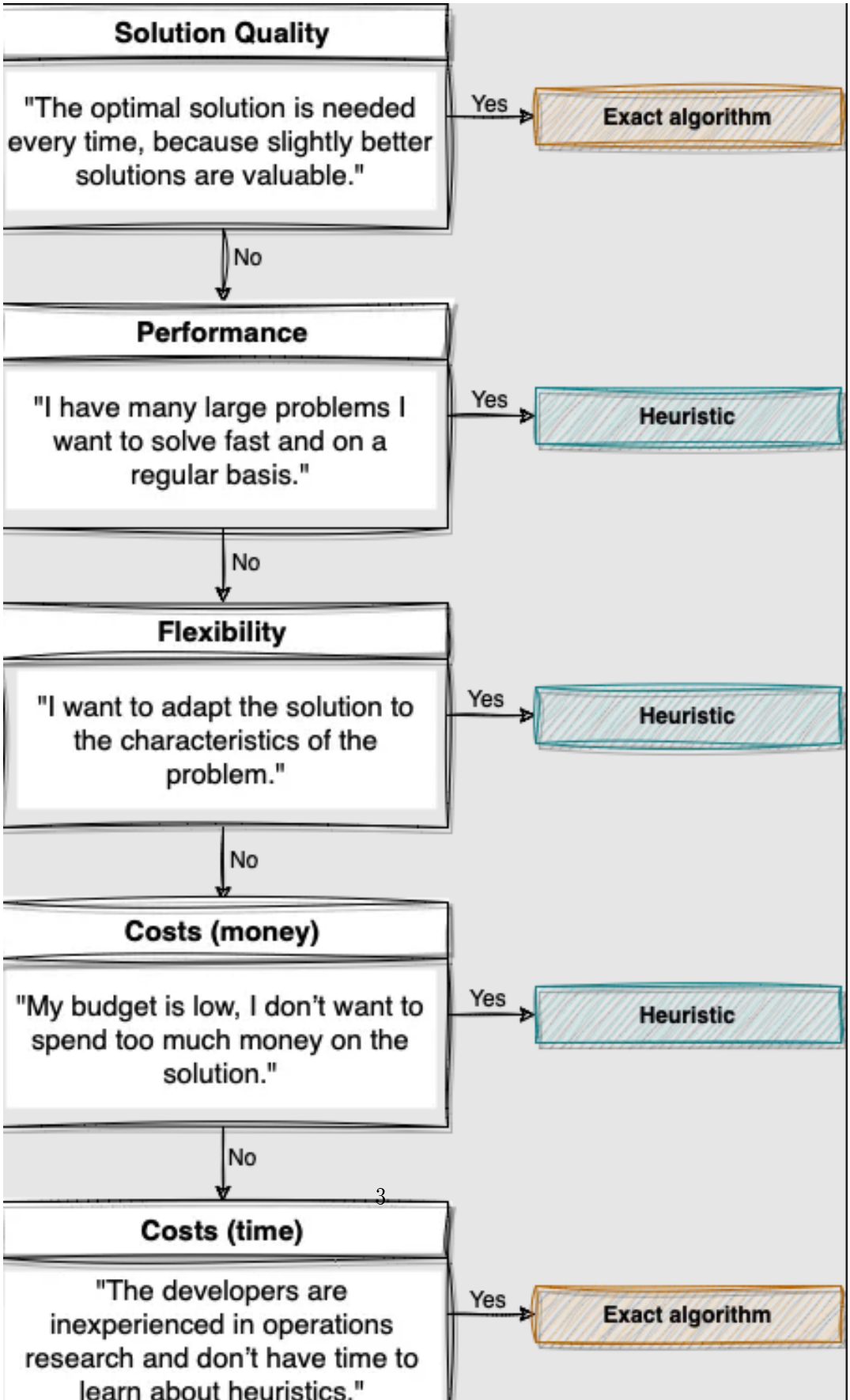
- The pseudocode **??** leads to the condition `health < 100`:

```
if health < 100
  Drink health potion
else
  Resume battle
end if
```

- Notice that you could also use another operator: `>=` This operator would have had the same effect but it is not what you were supposed to code. How would the pseudocode change with `health >= 100`?

[1] In Org mode, you can use the language as an example header argument to enable syntax highlighting. For pseudocode, this will of course not work perfectly, since most syntax elements are not in C.

[2] Undeclared variable `health`, missing closure semi-colons after the statements, functions `Drink` and `Resume` not known, and more.


Solution Quality
"The optimal solution is needed every time, because slightly better solutions are valuable."
Yes
Exact algorithm
No
Performance
"I have many large problems I want to solve fast and on a regular basis."
Yes
Heuristic
No
Flexibility
"I want to adapt the solution to the characteristics of the problem."
Yes
Heuristic
No
Costs (money)
"My budget is low, I don't want to spend too much money on the solution."
Yes
Heuristic
No
Costs (time)
"The developers are inexperienced in operations research and don't have time to learn about heuristics."
Yes
Exact algorithm

```
if health >= 100
  Resume battle
else
  Drink health potion
end if
```

- **Rule:** when making models (via **abstraction**), always stay as close to the **problem** description as possible - in terms of **language**, **logic**, **tone**, etc If you're unsure, **ask** (your client/professor/colleague).
- In the pseudocode example above, what relates to 1) language, 2) logic, and 3) tone? And 4) are there other specifications?[3]

5 Let's practice!

Get the practice file from GitHub: `9_pseudocode_practice.org` as `bit.ly/pseudo_practice`

[3] 1) Language: words used like `health`, `drink` or `resume`. 2) Logic: IFTTT (If This Then That). 3) Tone: game language 'drink health potion' as function `drink_health_potion`. 4) The indentation and the use of `if else end if` instead of, e.g. "WHEN health $<$ 100 THEN drink health potion OTHERWISE resume battle".