Computer Science Team Week 16

Henry Gustafson Julian Bauer

The College Preparatory School

Computer Science Team
May 6, 2024

ACSL 4

Topics are Graph Theory, Digital Electronics, Assembly Language.

Fun coding problems

Theme: Poetry

Problem Circuit

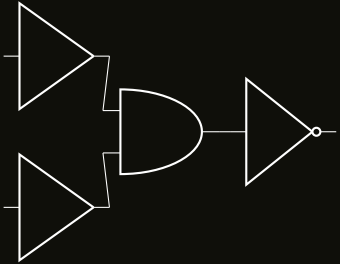
Problem Circuit You are creating a poetic circuit. Given *nodes*, a list of strings from the names of the ACSL digital electronics table, *edges*, a list of tuples (i, j) connecting nodes[i] $\to$ nodes[j] by wire, and *inputs*, a list of outputs (each is 0 or 1) for the first *n* nodes (where *n* is len(*inputs*)), output the output of the last node. If a node that requires 2 inputs doesn't have enough, it outputs 0.

```python
def poetic_circuit(
    nodes: list[str],
    edges: list[tuple[int, int]],
    inputs: list[int]
) -> int
```

Problem Circuit

Example

The circuit looks like this:



```
assert poetic_circuit(
    ["BUFFER", "BUFFER", "AND", "NOT"],
    [(0, 2), (1, 2), (2, 3)],
    [0, 1],
) == 1
```

Problem Assembly

Problem Assembly Joshua has written a poem made of op codes from the ACSL assembly op codes table. However, Joshua wants to simplify his rhyming pattern to AAAAAAAA... (all the lines must rhyme). Given *lines*, a list lines made of op codes separated by spaces and punctuation, Joshua wants to find the minimum number of words that need to be replaced to achieve his rhyme scheme. *Notes:* For two lines to rhyme, the pronunciation of the last syllable, excluding the leading consonants, must match exactly. The op codes are pronounced like english words, except for BG, BL, BU, and DC—those are pronounced by saying each letter (e.g. BG = bee jee). READ can be pronounced as red or reed, whichever minimizes the number of words you need to change.

```
def op_poem(
    lines: list[str]
) -> int
```

Problem Assembly

Example

```
assert op_poem([
    "PRINT STORE BE;",
    "BU, END PRINT STORE.",
    "PRINT STORE LOAD SUB;"
    "SUB DIV, END DC."
]) == 2 # you could change STORE to BG and SUB to DC.
```

Problem Abecedarian

Problem Abecedarian An Abecedarian is a type of poem where first letters of each line make the alphabet. For example,

”A programming language that both a cra-
b and the best is of
course the gol-
den Rust, with its blazingly fast...”

Would be the beginning of a good Abecedarian. Given a list of all words in english, return the minimum wordlist needed to create a full Abecedarian (a word can span multiple lines, see the above example).

```
def min_abecedarian(
    words: list[str]
) -> list[str]
```

Problem Abecedarian

Example Words taken from scrabble words Github.

```
assert min_abecedarian(
    # the words from https://raw.githubusercontent.com/
    # scrabblewords/scrabblewords/main/words/
    # North-American/NWL2020.txt
) == [
    "abaca", "defog", "hijack",
    "lymphadenopathy", "querist", "purview",
    "oxygenize"
]
```

The End

Questions? Comments? Remarks?
Considerations? Confusions?