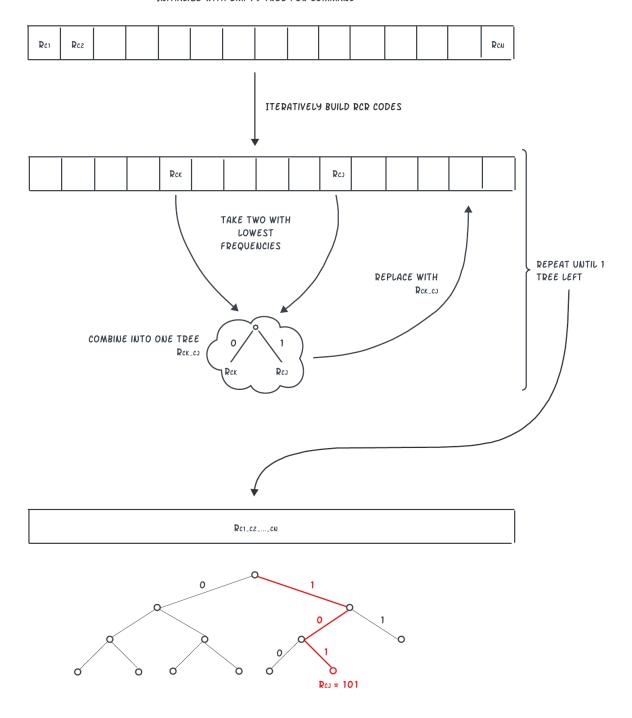# Robotic Code Representation

Our research lab has designed a robot that can execute various commands, e.g. `LEFT`, `DOWN`, `GRAB`, `UP`, `RIGHT`, `DROP`. Unfortunately, the commands need to be represented as strings composed of zeros and ones (so called Robotic Command Representation - RCR).

The robot is imperfect yet and processes long RCRs quite slowly. We would like to generate short RCRs for commands that are used frequently (we have a log of all commands that have ever been issued).

Our algorithmic team has sketched an algorithm for creating the RCRs:

INITIALIZE WITH EMPTY TREE PER COMMAND

R_{C1} R_{C2} R_{CN}

ITERATIVELY BUILD RCR CODES

R_{CK} R_{CJ}

TAKE TWO WITH LOWEST FREQUENCIES

REPLACE WITH R_{CK_CJ}

REPEAT UNTIL 1 TREE LEFT

COMBINE INTO ONE TREE R_{CK_CJ}

0 1

R_{CK} R_{CJ}

$R_{C1\_C2\_\ldots\_CN}$

0 1

0 1

0 1

R_{CJ} = 101

Example

For example, for the given log of commands: LEFT, GRAB, LEFT, BACK, LEFT, BACK, LEFT the generated RCRs would be:

- `00` for `GRAB`
- `01` for `BACK`
- `1` for `LEFT`

Task

Implement a python class RoboticCodeRepresentationGenerator with:

- a constructor with the following signature

```
def __init__(self, issued_commands: List[str])
```

  where `issued_commands` is the log of all commands ever issued,
- a method that calculates `rcr` for the given command:

```
def get_rcr(self, command: str) -> str
```

Please do not use any third party dependencies. You are of course free to use anything you need from the python standard library or implement any auxiliary methods or classes.