ex_02

Nikolai German

2025-05-05

Exercise 2: Binary to Numeric Conversion (2 Complement)

```r
binaryToNumber <- function(binary) {
  checkmate::assertIntegerish(binary, lower = 0, upper = 1, any.missing = FALSE, min.len = 2)
  res <- 0
  n <- length(binary)
  for (i in seq_along(binary)) {
    res <- res + (1 - (2 * (i == 1))) * binary[[i]] * 2^(n-i)
    # if (i == 1) {
    #   res <- res - binary[[i]] * 2^(n-1)
    # } else {
    #   res <- res + binary[[i]] * 2^(n-i)
    # }
  }
  return(res)
}

binaryToNumber(c(0,0,1,1,0,1,1,1))
```

```
## [1] 55
```

```r
binaryToNumber(c(1,1,1,1,1,0,0,0))
```

```
## [1] -8
```

```r
binaryToNumber(c(0,0,0,0,1,0,0,0))
```

```
## [1] 8
```

```r
binaryToNumber(c(1, 1, 0, 1))
```

```
## [1] -3
```

```r
binaryToNumber(c(0, 0, 0, 1, 0, 1, 1))
```

```
## [1] 11
```

Exercise 4: Character Encoding

(a) Implement UTF-8 Conversion Functions

```r
toBits <- function(num, n.bits) {
  checkmate::assertIntegerish(n.bits,
                              lower = 1,
                              len = 1,
                              any.missing = FALSE)
  checkmate::assertIntegerish(num,
                              lower = 0,
                              upper = 2^n.bits - 1,
                              len = 1,
                              any.missing = FALSE)
  res <- integer(n.bits)
  for (i in seq_len(n.bits)) {
    res[[i]] <- num %/% (2^(n.bits - i))
    num <- num %% (2^(n.bits - i))
  }
  return(res)
}

toNumber <- function(Bits) {
  checkmate::assertIntegerish(Bits,
                              lower = 0,
                              upper = 1,
                              min.len = 1,
                              any.missing = FALSE)
  n <- length(Bits)
  res <- 0
  for (i in seq_along(Bits)) {
    res <- res + Bits[[i]] * 2^(n-i)
  }
  return(res)
}
```

```r
toBits(15, 8) |> toNumber()
```

```
## [1] 15
```

```r
codepointsToUtf8 <- function(codepoints) {
  checkmate::assertIntegerish(codepoints,
                              lower = 0,
                              upper = (2^20 + 2^16 - 1),
                              min.len = 1,
                              any.missing = FALSE)

  utf8 <- list()

  for (i in seq_along(codepoints)) {
    if (codepoints[[i]] < 2^7) {
      utf8[[i]] <- codepoints[[i]]
    } else if (codepoints[[i]] < 2^11) {
      bits <- toBits(codepoints[[i]], n.bits = 11)
      utf8[[i]] <- c(toNumber(c(1, 1, 0, bits[1:5])),
                     toNumber(c(1, 0, bits[6:11])))
    } else if (codepoints[[i]] < 2^16) {
      bits <- toBits(codepoints[[i]], n.bits = 16)
      utf8[[i]] <- c(toNumber(c(1, 1, 1, 0, bits[1:4])),
                     toNumber(c(1, 0, bits[5:10])),
                     toNumber(c(1, 0, bits[11:16])))
    } else {
      bits <- toBits(codepoints[[i]], n.bits = 21)
      utf8[[i]] <- c(toNumber(c(1, 1, 1, 1, 0, bits[1:3])),
                     toNumber(c(1, 0, bits[4:9])),
                     toNumber(c(1, 0, bits[10:15])),
                     toNumber(c(1, 0, bits[16:21])))
    }
  }
  return(unlist(utf8))
}
```

```r
codepointsToUtf8(c(127,
                   128,
                   2047,
                   2048,
                   2^16-1,
                   2^16,
                   2^20+2^16 - 1))
```

```
##  [1] 127 194 128 223 191 224 160 128 239 191 191 240 144 128 128 244 143 191 191
```

```r
utf8ToCodepoints <- function(bytes) {
  checkmate::assertIntegerish(bytes,
                              lower = 0,
                              upper = 255,
                              min.len = 1,
                              any.missing = FALSE)

  n <- length(bytes)
  i <- 1
  l <- 1
  res <- list()

  while (i <= n) {
    leading.byte <- bytes[[i]]
    if (leading.byte < 192) {
      res[[l]] <- leading.byte
      i = i + 1
    } else if (leading.byte < 224) {
      temp <- vapply(bytes[i:(i + 1)],
                     function(x) toBits(x, n.bits = 8),
                     numeric(8))
      res[[l]] <- toNumber(c(temp[4:8, 1], temp[3:8, 2]))
      i = i + 2
    } else if (leading.byte < 240) {
      temp <- vapply(bytes[i:(i + 2)],
                     function(x) toBits(x, n.bits = 8),
                     numeric(8))
      res[[l]] <- toNumber(c(temp[5:8, 1], temp[3:8, 2], temp[3:8, 3]))
      i = i + 3
    } else {
      temp <- vapply(bytes[i:(i + 3)],
                     function(x) toBits(x, n.bits = 8),
                     numeric(8))
      res[[l]] <- toNumber(c(temp[6:8, 1], temp[3:8, 2], temp[3:8, 3], temp[3:8, 4]))
      i = i + 4
    }
    l <- l + 1
  }
  return(unlist(res))
}
```

```
codepointsToUtf8(c(127,
                   128,
                   2047,
                   2048,
                   2^16-1,
                   2^16,
                   2^20+2^16 - 1)) |> utf8ToCodepoints()
```

```
## [1]     127     128    2047    2048   65535   65536 1114111
```

(b) Explore Character Representation

```
bytes <- readBin("../data/chars.txt", "integer", n = 100, size = 1, signed = FALSE)

codepoints <- utf8ToCodepoints(bytes)

sprintf("U+%04X", codepoints)
```

```
## [1] "U+0058"  "U+00E4"  "U+1F525" "U+1F44D" "U+1F469" "U+200D"  "U+1F467"
```