no_debruijn

May 28, 2020

```
In [ ]: def no_duplicates(s,n):
            m = len(s)
            for i in range(m-2*n):
                for j in range(i+n,m-n):
        #            print s[i:i+n], s[j:j+n]
                    if s[i:i+n] == s[j:j+n]:
                        return False
            return True
```

```
In [ ]: print no_duplicates('00000100011001010011101011001011111',5)
```

```
In [ ]: print no_duplicates('0000000000100011001010011101011011111',5)
```

```
In [36]: def duplicatable(s,n):
             print s
             m = len(s)
             for i in range(m):
                 for j in range(i+1,min(i+n,m+1)):
                     new_s = s[:i] + s[i:j] + s[i:]
                     if no_duplicates(new_s,n):
                         print s[i:j], i, j
```

```
In [37]: s = DeBruijnSequences(2,7).an_element()
         sWd = Word(s)
         duplicatable(sWd,7)
```

```
00000001000001100001010000111000100100010110001101000111100100110010101001011100110110011101001
0 0 1
00 0 2
000 0 3
0000 0 4
00000 0 5
000000 0 6
0 1 2
00 1 3
000 1 4
0000 1 5
00000 1 6
```

000000 1 7
0 2 3
00 2 4
000 2 5
0000 2 6
00000 2 7
000001 2 8
0 3 4
00 3 5
000 3 6
0000 3 7
000010 3 9
0 4 5
00 4 6
000 4 7
000100 4 10
0 5 6
00 5 7
001000 5 11
0 6 7
010000 6 12
100000 7 13
000001 8 14
10101 100 105
01011 101 106
10110 102 107
01101 103 108
11010 104 109
10101 105 110
01011 106 111
1101 114 118
1011 115 119
0111 116 120
1110 117 121
1101 118 122
1011 119 123
0111 120 124
1 121 122
11 121 123
111 121 124
1111 121 125
11111 121 126
111111 121 127
1 122 123
11 122 124
111 122 125
1111 122 126
11111 122 127

```
111111 122 128
1 123 124
11 123 125
111 123 126
1111 123 127
11111 123 128
1 124 125
11 124 126
111 124 127
1111 124 128
1 125 126
11 125 127
111 125 128
1 126 127
11 126 128
1 127 128
```

```
In [38]: s = DeBruijnSequences(2,8).an_element()
         sWd = Word(s)
         duplicatable(sWd,8)
```

```
0000000010000001100000101000001110000100100001011000011010000111100010001001100010101000010111000
0 0 1
00 0 2
000 0 3
0000 0 4
00000 0 5
000000 0 6
0000000 0 7
0 1 2
00 1 3
000 1 4
0000 1 5
00000 1 6
000000 1 7
0000000 1 8
0 2 3
00 2 4
000 2 5
0000 2 6
00000 2 7
000000 2 8
0000001 2 9
0 3 4
00 3 5
000 3 6
0000 3 7
```

```
00000 3 8
0000010 3 10
0 4 5
00 4 6
000 4 7
0000 4 8
0000100 4 11
0 5 6
00 5 7
000 5 8
0001000 5 12
0 6 7
00 6 8
0010000 6 13
0 7 8
0100000 7 14
1000000 8 15
0000001 9 16
1000 64 68
0001 65 69
0010 66 70
0100 67 71
1000 68 72
0001 69 73
0010 70 74
0100 71 75
1100 163 167
1001 164 168
0011 165 169
0110 166 170
1100 167 171
1001 168 172
0011 169 173
0110 170 174
10 208 210
1010 208 212
101010 208 214
01 209 211
0101 209 213
010101 209 215
10 210 212
1010 210 214
101010 210 216
01 211 213
0101 211 215
010101 211 217
10 212 214
1010 212 216
```

```
01 213 215
0101 213 217
10 214 216
01 215 217
1110 240 244
1101 241 245
1011 242 246
0111 243 247
1 244 245
1110 244 248
1 245 246
1101 245 249
1 246 247
1011 246 250
0111 247 251
1 248 249
11 248 250
111 248 251
1111 248 252
11111 248 253
111111 248 254
1111111 248 255
1 249 250
11 249 251
111 249 252
1111 249 253
11111 249 254
111111 249 255
1111111 249 256
1 250 251
11 250 252
111 250 253
1111 250 254
11111 250 255
111111 250 256
1 251 252
11 251 253
111 251 254
1111 251 255
11111 251 256
1 252 253
11 252 254
111 252 255
1111 252 256
1 253 254
11 253 255
111 253 256
1 254 255
```

```
11 254 256
1 255 256
```

In [ ]: