KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Programavimo kalbų teorija (P175B124)

Laboratorinių darbų ataskaita

Atliko:
IFF-0/3 gr. studentė
Simona Ragauskaitė
2022 m. balandžio 11 d.
Priėmė:
Lekt. Evaldas Guogis
Doc. Svajūnas Sajavičius

KAUNAS 2022

TURINYS

1. Haskell (LD3)

1.1. Darbo užduotis

729 užduotis

The Hamming distance between two strings of bits (binary integers) is the number of corresponding bit positions that differ. This can be found by using XOR on corresponding bits or equivalently, by adding corresponding bits (base 2) without a carry. For example, in the two bit strings that follow:

```
      A       0 1 0 0 1 0 1 0 0 0
      B       1 1 0 1 0 1 0 1 0 0
A XOR B = 1 0 0 1 1 1 1 1 0 0
```

The Hamming distance (H) between these 10-bit strings is 6, the number of 1's in the XOR string.

Input

Input consists of several datasets. The first line of the input contains the number of datasets, and it's followed by a blank line. Each dataset contains N, the length of the bit strings and H, the Hamming distance, on the same line. There is a blank line between test cases.

Output

For each dataset print a list of all possible bit strings of length N that are Hamming distance H from the bit string containing all 0's (origin). That is, all bit strings of length N with exactly H 1's printed in ascending lexicographical order.

The number of such bit strings is equal to the combinatorial symbol $C(N, H)$. This is the number of possible combinations of $N - H$ zeros and H ones. It is equal to

$$\frac{N!}{(N-H)!H!}$$

This number can be very large. The program should work for $1 \leq H \leq N \leq 16$.
Print a blank line between datasets.

1.2. Programos kodas

```
countOnes ::String -> Integer
countOnes str = toInteger . length $ filter (\x -> x == '1') str

perms ::Integer -> [String]
perms sk
  | sk == 0 = [""]
  | otherwise = concat $ map (\x -> [x++"0", x++"1"]) $ perms $ sk-1

filteredPerms ::  Integer -> Integer -> [String]
filteredPerms sk cnt = filter (\x -> countOnes(x) == cnt) $ perms sk

main = do
  s <- readFile "duom.txt"
  writeFile "res.txt" $ concat
    $ map (\x -> unlines [unlines $ filteredPerms (x!!0) (x!!1)] )
    $ map (\x -> map (\y -> read y :: Integer) $ words x) $ tail $ lines s
```

1.3. Programos testavimas

1 variantas
duom.txt

```
1
4 2
```

rez.txt

```
0011
0101
0110
1001
1010
1100
```

2 variantas

duom.txt

```
1
3 2
```

rez.txt

```
011
101
110
```