PYTHON

Osnove 11

Python urejevalnik se nahaja na strani:
`https://www.w3resource.com/python-exercises/python-basic-exercises.php#EDITOR`
Naloge Kattis:`https://github.com/minidomo/Kattis`

1. **Faktorizacija**
   Naloga
   Za dano naravno število poiščite praštevila, katerih produkt je to število.
   Praštevilo je naravno število, ki je deljivo le z 1 in samim seboj. Števila 1 ne prištevamo med praštevila. Tako razcep števila na produkt praštevil postane enoličen.
   Vhod:
   Iz standardnega vhoda preberete število.
   Izhod:
   Na standardni izhod izpišete praštevila, urejena v naraščajočem vrstnem redu, ločena s presledkom, katerih produkt je dano število.
   Primeri:

   ```
   vhod: 144  izhod: 2 2 2 2 3 3
   vhod:  17  izhod: 17
   ```

2. **Goldbachova domneva**

   Vaša naloga je najti vse predstavitve danega sodega števila kot vsote dveh praštevil. Praštevilo je celo število, večje od 1, ki je deljivo le samo s seboj in z 1. Prvih nekaj praštevil je

   $$2, 3, 5, 7, 11, 13, 17, 19, 23, 31, \ldots.$$

   Nemški matematik Christian Goldbach (1690–1764) je domneval, da je mogoče vsako sodo število večje od 2, predstaviti kot vsoto dveh praštevil. (Ta domneva ni bila nikoli dokazana niti nikoli ni bil najden protiprimer.)

   Dano sodo število lahko predstavimo z vsoto praštevil na več načinov. Na primer število 28 lahko predstavljeno kot $28 = 5 + 23$ ali kot $28 = 11 + 17$.

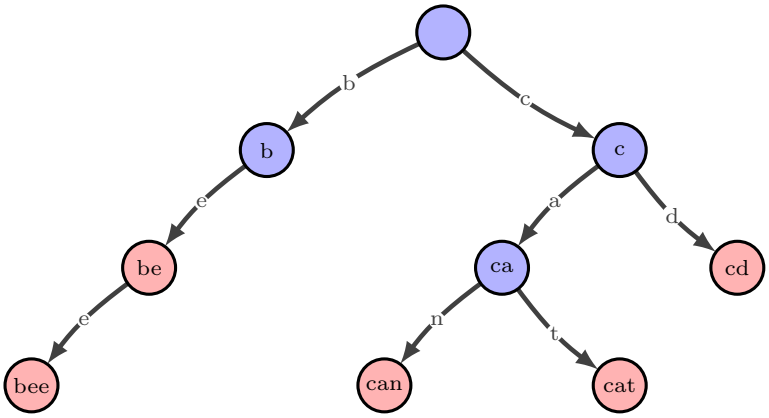
   Iz standardnega vhoda preberete število `n` in na standardni izhod izpišete pare praštevil, katerih vsota je dano število.

   Primer

   ```
   Vhod: 34
   Izhod: [(3, 31), (5, 29), (11, 23), (17, 17)]
   ```

3. **Ugibanje števila**
   `https://scratch.mit.edu/projects/223021286/editor`

4. **Trie**
   Spomnimo se aplikacije na mobilnih telefonih, ki nam, ko začnemo tipkati, začne prikazovati mogoča nadaljevanja besede že po nekaj vnesenih znakih. Aplikacija je zelo priročna, ker nam omogoča, da vpisujemo besede, ki so pravilno črkovane. Kako računalnik v vašem prenosnem telefonu lahko poišče mogoča nadaljevanja besed, ki jih vnašate? Slovar v vašem računalniku je zapisan v posebni podatkovni strukturi, ki se imenuje `trie`.



Naloga:
Za dani seznam besed zgradimo drevo `trie`.
Vhod:
Iz standardnega vhoda preberemo seznam besed.
Izhod:
Na standardni izhod izpišemo slovar, ki predstavlja drevo `trie`.
Primeri:

```
1.
vhod: ['be', 'bee', 'can', 'cat', 'cd']
izhod: {"b": {"e": {"$": {},
                    "e": {"$": {}}}},
        "c": {"a": {"n": {"$": {}},
                          "t": {"$": {}}},
              "d": {"$": {}}}}

2.
vhod: ["jablana","jabolko", "hruška"]
izhod:
{"j": {"a": {"b": {"o": {"l": {"k": {"o": {"$": {}}}}},
                   "l": {"a": {"n": {"a": {"$": {}}}}}}}},
 "h": {"r": {"u": {"s": {"k": {"a": {"$": {}}}}}}}}
```

Rešitve 11

Faktorizacija

```python
#!/usr/bin/env python3
from math import sqrt

def factorize(n):
    sieve = [True] * (n + 1)

    for x in range(2, int(len(sieve) ** 0.5) + 1):
        if sieve[x]:
            for i in range(x + x, len(sieve), x):
                sieve[i] = False

    lowerPrimes = [i for i in range(2, len(sieve)) if sieve[i] and (n % i == 0)]
    return lowerPrimes

def factors(n):    # (cf. https://stackoverflow.com/a/15703327/849891)
    j = 2
    while n > 1:
        for i in range(j, int(sqrt(n + 0.05)) + 1):
            if n % i == 0:
                n //= i ; j = i
                yield i; break
        else:
            if n > 1:
                yield n; break

if __name__ == '__main__':
    n = int(input("n = "))
    print(factorize(n))
    print(list(factors(n)))
```

Godbachova domneva

```python
#!/usr/bin/env python3

def primes(n):
    l = range(2, n + 1)
    p = []
    while l:
        p.append(l[0])
        l = [x  for x in l if (x % l[0] != 0)]
    return p

def goldbach(n):
    p = primes(n)
    pair = []
    print(p)
    for i in range(len(p)):
        for j in range(i, len(p)):
            if p[i] + p[j] == n:
                pair.append((p[i],p[j]))
    return pair


if __name__ == '__main__':
    n = int(input('n -> '))
    print(goldbach(n))
```

Ugibanje števila

```
#!/usr/bin/env python3

def dec_bin(n):
    res = []
    while n:
        res += [n % 2]
        n //= 2
    return res

def tables(m):
    t = [[] for x in range(m)]
    for i in range(2**m):
        ib = dec_bin(i)
        ib += [0] * (m - len(ib))
        for j in range(len(ib)):
            if ib[j]:
                t[j].append(i)
    return t

if __name__ == '__main__':
    t = tables(6)
    n = 0
    for i in range(len(t)):
        print(t[i][:16])
        print(t[i][16:])
        p = input('+/- ')
        if p == '+':
            n += 2**i
    print(n)
```

Trie

```python
#!/usr/bin/env python3
import json

class Trie(object):
    def __init__(self):
        self.child = {}

    def insert(self, word):
        current = self.child
        for l in word:
            current[l] = current.get(l, {})
            current = current[l]
        current['#'] = {}

    def search(self, word):
        current = self.child
        for l in word:
            if l not in current:
                return False
            current = current[l]
        return '#' in current

    def start_with(self, prefix):
        current = self.child
        for l  in prefix:
            if l not in current:
                return false
            current = current[l]
        return True


    def __str__(self): return json.dumps(self.child)

if __name__ == '__main__':
    ob = Trie()
    ob.insert('jabolko')
    ob.insert('jablana')
    ob.insert('hruska')
    #print(ob.start_with('ja'))
    #print(ob.search('jabla'))
    #print(ob.search('jabolko'))
    print(ob)
```