Optimal search for rationals

Matej Cerar

21. 5. 2024

Predstavitev problema

- Kako optimalno poiskati racionalno število v seznamu oblike
  $O_M = \{p/q \mid p, q \in \{1, 2, \cdots, M\}\}$, le z vprašanjem ali je izbrano število manjše ali večje od x?

Pregled reševanja

1. Eksponentno iskanje intervala.
2. Binarno iskanje celega dela.
3. Isakanje dovolj majhnega ulomka.
4. iskanje decimalnega dela.

Eksponentno iskanje celega dela

```
def integer_part_interval(n):
    if n < 1:
        return ([[0,1]])
    else:
        i= 1
        seznam = [[0,1]]
        while i <= n:
            seznam.append(generiraj_seznam_od_do([i,2*i]))
            i = 2*i
        return seznam
```

Slika 1: Coda za eksponentno iskanje

Iskanje celega dela

```python
def integer_part_solution_1(sez,n,i = None, sez_vseh_intervalov = None):
    #print("Vstopili smo v funkcijo")
    if i is None:
        i = 0
    if sez_vseh_intervalov is None:
        sez_vseh_intervalov = []

    if len(sez) == 0 :
        return (i, -1)
    min = sez[0]
    #print("min", min)
    max = sez[-1]
    #print("max", max)
    if len(sez) == 1:
        #print("Konec, ker je dolžina seznama 1")
        if min == n:
            sez_vseh_intervalov.append([n])
            return (i, sez_vseh_intervalov)
        else:
            return -1 #if number is not in a list
    if n == floor((max+min)/2):
        #print("Konec, ker smo zadeli število")
        sez_vseh_intervalov.append([n])
        return (i, sez_vseh_intervalov)
```

Slika 2: iskanje celega dela

```python
    elif floor((max+min)/2)  < n:
        sez1 = sez[sez.index(ceil((max+min)/2)):]
        #print("Vzeli smo drugo polovico", sez1)
        sez_vseh_intervalov.append(sez1)
        i += 1
        #print("število iteracije", i)
        return integer_part_solution_1(sez1,n,i,sez_vseh_intervalov)

    else:
        sez2 = sez[:sez.index(floor((max+min)/2)+1)]
        #print("Vzeli smo prvo polovico", sez2)
        sez_vseh_intervalov.append(sez2)
        i += 1
        #print("število iteracij", i)
        return integer_part_solution_1(sez2,n,i,sez_vseh_intervalov)
```

Slika 3: Iskanje celega dela

Lema 4 in iskanje intervala za decimalni del

```python
def binary_search_for_fraction(lower,upper, x, M,i= None, seznam_vseh_intervalov = None):
    if i is None:
        i = 0
    if seznam_vseh_intervalov is None:
        seznam_vseh_intervalov = []
    if floor(x) == 0:
        m = M
    else:
        m = floor(M/ floor(x))
    if x <= (lower+ upper)/2:
        # torej x v prvi polovici intervala, treba preveriti ali interval že dovolj majhen
        #print( "Vzamemo prvo polovico intervala")
        seznam_vseh_intervalov.append([Fraction(lower), Fraction((lower+ upper)/2)])
        if (lower+ upper)/2 - lower < 1/(2*(m**2)):
            return ((Fraction(lower), Fraction((lower+ upper)/2)), i, seznam_vseh_intervalov)
        else:
            i += 1
            #print(i)
            return binary_search_for_fraction(Fraction(lower), Fraction((lower+ upper)/2), x, M, i, seznam_vseh_intervalov)
    else:
        # torej x v drugi polovici intervala, postopamo podobno
        #print( "Vzamemo drugo polovico intervala")
        seznam_vseh_intervalov.append([Fraction((lower+ upper)/2), Fraction(upper)])
        if  upper - (lower+ upper)/2 < 1/(2*(m**2)):
            return ((Fraction((lower+ upper)/2), Fraction(upper)), i, seznam_vseh_intervalov)
        else:
            i += 1
            #print(i)
            return binary_search_for_fraction(Fraction((lower+ upper)/2), Fraction(upper), x, M, i, seznam_vseh_intervalov)
```

Slika 4: Iskanje intervala za decimalni del

Lema 5 in določitev iskane številke

```
def find_fraction(a, b, c, d):
    if floor(a/b) == floor(c/d) and (a / b) % 1 != 0:  # Case 1
        b_, aa = find_fraction(d, c % d, b, a % b)
        #print( b_, aa)
        a_ = floor(a/b) * b_ + aa
        return Fraction(a_, b_)
    else:  # Case 2
        return ceil(a/b), 1
```

Slika 5: Iskanje izbrane številke

Spletna stran

Implementacija spletne strani in predstavitev delovanja.