Problemi deljivosti, dinamičko programiranje – Python 3

Deljiv 2-3-5

Vremensko ograničenje	Memorijsko ograničenje	ulaz	izlaz
0,1 s	64 MB	standardni ulaz	standardni izlaz

Posmatrajmo niz brojeva čiji su prosti činioci samo 2, 3 i 5 (svaki može da se javi nula i više puta). To su brojevi 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, ... Napiši program koji određuje n-ti član ovog niza (brojanje kreće od 0). Niz brojeva zovemo Vivaldijev niz.
Ulaz

Sa standardnog ulaza se učitava broj n (0≤n≤10000).
Izlaz

Na standardni izlaz ispisati traženi n-ti član niza.
Primer
Ulaz
500
Izlaz
944784

IDEJA
Trazeni broj se može predstaviti kao x = 2^x*3^y*5^z
Svaki clan niza se može predstaviti kao proizvod prethodnih clanova niza.

Potreban nam je niz od 10000 brojeva.

Koristićemo klasičnu Python listu zbog uštede vremena, jer moramo da iteriramo kroz nju. U C++ bi koristili deque.

Procena: naša lista sadrži 10000 64-bitnih vredosti.

Ideja algoritma: idemo iterativno po listi. Traženi brojevi se dobijaju množenjem prethodnih članova niza.
Manje efikasno resenje: Imaćemo dva iteratora koji množe prethodne članove niza sa 2 i 3 i uporeduju dve vrednosti koje su sledece po vrednosti. jedan iterator u tom slucaju nastavlja dalje, a drugi ostaje na istom mestu. Problem ostaje sa potpunim stepenima petice.

Efikasnije resenje: Imaćemo dva iteratora koji množe prethodne članove niza sa 2 i 3 i uporeduju dve vrednosti koje su sledece po vrednosti. Jedan iterator u tom slucaju nastavlja dalje, a drugi ostaje na istom mestu. Problem sa potpunim stepenima petice ćemo rešiti tako što cemo u posebnoj vrednosti cuvati sledeci stepen petice. Ako je on manji od obe vrednosti, ubacujemo njega u niz.
Algoritam ce imati složenost O(n) zato što imamo dva jedno-dimenziona iteratora.

```
vivaldi = [1]

a = b = c = 0
i=0
n=int(input())
while i<n:
    sledeci = min(2*vivaldi[a], 3*vivaldi[b], 5*vivaldi[c])
    vivaldi.append(sledeci)
    if sledeci == 2*vivaldi[a]: a += 1
    if sledeci == 3*vivaldi[b]: b += 1
    if sledeci == 5*vivaldi[c]: c += 1
    i+=1
```

```
print (vivaldi[n])
```

C++ rešenje

```
#include <iostream>
#include <queue>
#include <algorithm>

using namespace std;

int main() {
  int n;
  cin >> n;

  deque<unsigned long long> niz2, niz3, niz5;

  niz2.push_back(2); niz3.push_back(3); niz5.push_back(5);
  unsigned long long vivaldi = 1;

  for (int i = 0; i < n; i++) {
    vivaldi = min({niz2.front(), niz3.front(), niz5.front()});
    niz2.push_back(2*vivaldi);
    niz3.push_back(3*vivaldi);
    niz5.push_back(5*vivaldi);
    while (niz2.front() == vivaldi) niz2.pop_front();
    while (niz3.front() == vivaldi) niz3.pop_front();
    while (niz5.front() == vivaldi) niz5.pop_front();
  }
  cout << vivaldi << endl;
  return 0;
}
```

2. Sazimanje K uzastopnih jednakih slova

Vremensko ograničenje	Memorijsko ograničenje	ulaz	izlaz
1 s	64 MB	standardni ulaz	standardni izlaz

Data je niska koju čini N malih slova engleske abecede. Nad niskom definišemo operaciju Black Hole Sun na sledeći način: uklanjanje serije od K uzastopnih jednakih slova. Ako nad datom ulaznom niskom primenite operaciju Black Hole Sun dokle god je moguće, napisati program koji će ispisati rezultujuću nisku. (Rezultujuća niska je jedinstvena.)

Ulaz

U prvom redu standardnog ulaza nalaze se dva broja N, K ($1 \leq K \leq N \leq 100000$). U drugom redu standardnog ulaza data je polazna niska koja se transformiše na opisani način.

Izlaz

Na standardni izlaz ispisati traženu nisku.

Primer

Ulaz

9 4

abbbccccb

Izlaz

a

IDEJA
Употребимо стек који на врху памти текући карактер из улазне ниске и број појава.
Када наиђе текући карактер у нисци, пореди се са врхом стека.
Ако је врх стека једнак карактеру, увећа се број појава.
(и евентуално скине врх стека ако је број појава једнак к).
Иначе, се поставља нови карактер на стек.
На крају се испише садржај са стека.

C++ rešenje

```
#include <algorithm>
#include <iostream>
#include <deque>
using namespace std;
int main() {
    int n, k;
    string str;
    cin >> n >> k >> str;

            // Pravimo stack stk kao deque, da na kraju kad stampamo rezultujucu jedinstvenu nisku
            //ne moramo da obrcemo nisku
    deque< pair<char, int> > stk;

            // jednim prolazom kroz ulaznu nisku vrsimo obradu
    for (int i = 0; i < n; ++i) {

        if (!stk.empty() && stk.back().first == str[i])
  // ako se tekuci element ulazne niske poklapa sa vrhom steka, uvecajmo broj pojava tog karaktera
            stk.back().second += 1;
         else  // u suprotnom, tekuci karakter ide na vrh steka
            stk.push_back({str[i], 1});
        if (stk.back().second == k)  // Ako poslednje malo slovo na steku ima broj pojava jednak sa k,
uklonimo vrh steka
            stk.pop_back();
    }

    while (!stk.empty()) {  // stampamo rezultat
        cout << string(stk.front().second, stk.front().first);
        stk.pop_front();
    }
    cout << endl;
    return 0;
}
```

Python resenje 1 (pravimo stack)

```
class Stack:

    def __init__(self):
        self.items=[]
    def isEmpty(self):
        return self.items==[]
    def push(self, item):
        self.items.append(item)
    def pop(self):
        return self.items.pop()
    def peek(self):
        return self.items[len(self.items)-1]
```

```
    def size(self):
        return len(self.items)

n,k=input().split(" ")
n=int(n)
k=int(k)
s=input()
stack=Stack()
for c in s:
    if stack.isEmpty():
        stack.push([c,1])
    else:
        if stack.peek()[0]==c:
            stack.items[stack.size()-1][1]+=1
        else:
            stack.push([c,1])
    if stack.items[stack.size()-1][1]==k:
        stack.pop()
s=""
for a in stack.items:
    s+=a[0]*a[1]
print(s)
```

Python resenje 2 (set)

```
import sys

nums = input()
nums = nums.split(" ")
n = int(nums[0])
k = int(nums[1])

if n < k:
    sys.exit(-1)

niska = input()

skup = set(niska)
for i in niska:
    niska = niska.replace(i*k, "")
print(niska)
```

3.

Deljivost i jos ponesto

Vremensko ograničenje	Memorijsko ograničenje	ulaz	izlaz
1 s	64 MB	standardni ulaz	standardni izlaz

Дат је скуп који садржи N природних бројева. Наћи највећи подскуп такав да за било који пар елемената A и B из тог подскупа или A дели B или B дели A.

Улаз

У првом реду стандардног улаза дат је цео број N ($1 \leq N \leq 2000$). У другој линији дато је N чланова скупа (све вредности су између 1 и 10^9).

Излаз

На стандарни излаз исписати један цео број који представља број чланова жељеног скупа.

Примери

Улаз	Излаз	Појашњење
5 9 36 17 108 3	4	Решење са 4 елемента је (9,36,108,3).
9 18 20 6 200 4 36 108 40 2	5	Решење са 5 елемената је (20,200,4,40,2).
10 16 108 8 360 6 756 18 2 36 34	6	Решење са 6 елемената је (108,6,756,18,2,36).

Решење:
Ако број ***A*** дели број ***B*** и број ***B*** дели број ***C*** онда број ***A*** дели број ***C***. (Докажите за вежбу) Транзитивност нам омогућава да извршимо једноставну проверу да видими да ли је неки подскуп ваљан у смислу ограничења описаног у задатку: сортирамо све бројеве у подскупу и проверимо да ли важи дељивост за узастопне бројеве.

То заправо значи да након сортирања морамо да нађемо најдужи подниз ***S*** тако да члан ***S[i]*** дели ***S[i+1]***. Користимо ДП технику да решимо задатак као што смо учили да решавамо сличан проблем налажења дужине највећег заједничког подниза.

C++ rešenje

```
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n; cin >> n;
    vector<int> skup(n), resenje(n, 0);
    for (int i = 0; i < n; i += 1) {
        cin >> skup[i];
    }

    sort(skup.begin(), skup.end());
    for (int i = 0; i < n; i += 1) {
        int najbolje_resenje = 0;
        for (int j = 0; j < i; j += 1) {
            if (skup[i] % skup[j] == 0 and najbolje_resenje < resenje[j]) {
                najbolje_resenje = resenje[j];
            }
        }
        resenje[i] = najbolje_resenje + 1;
    }

    cout << *max_element(resenje.begin(), resenje.end());
    return 0;
}
```

Python rešenje

```
n = int(input())

skup = list(reversed(sorted([int(x) for x in input().split()])))
```

```
dp = [0 for i in range(n)]
for i in range(n):
    for j in range(i):
        if skup[j] % skup[i] == 0 and dp[j] > dp[i]:
            dp[i] = dp[j]
    dp[i] += 1
print(max(dp))
```