Programazioaren Oinarriak

Ohiko Deialdia – 2025 Maiatza

Matematikako, Ingeniaritza Elektronikoko eta FIS+IEko Graduak – 1. Maila

Arauak

Ezin da apunterik, kalkulagailurik ez beste tresna elektronikorik erabili.

1. (**3 puntu**) Askotan erabiltzen ditugun teklatu iragarleei esker, idatzi beharreko karaktere kopurua minimizatu dezakegu. Halako gailuek hitz bat iragartzen dute tekleatutako lehenengo karaktereetatik (*aurrizkitik*) abiatuta. `a` aurrizki bat emanda, iragarria izango den hitza aurrizki horrekin hasten den `w` hitz probableena (ohikoena) izango da. Ondoko metodoak sortu:

   (a) `get_predict(h)`
   Hitz histograma bat adierazten duen `h` hiztegi bat (gakoa=hitza, balioa=agerpen_kopurua) jasotzen du eta hiztegi iragarle bat bueltatzen du (gakoa=aurrizkia, balioa=hitz_ohikoena). Hau da, emaitzako hiztegi iragarlea `pred` bada, `pred[a]` `a` aurrizkia duen hitz ohikoena da.

   (b) `saved_chars(w, pred)`
   `w` hitza eta `pred` hiztegi iragarlea jasotzen ditu eta hiztegi iragarleari esker `w` hitza idaztean aurrezten ditugun $n \in$ `[0, len(w)-1]` karaktere kopurua bueltatzen du. `w` hitza tekleatzean, `a` aurrizkiak sortzen hasiko gara eta horietako bakoitzarekin gure `pred` hiztegiak hitz bat iragarriko du. Iragarritako hitza `w` bada, `n=len(w)-len(a)` izango da aurreztutako karaktere kopurua. `a` hiztegi iragarlean ez badago, gure hitza osorik tekleatu beharko dugu, aurreztutako karaktere kopurua `0` izanik. Adibidez, `'eraman'` hitzarentzat ondokoa gerta liteke:

   ```
   a='e'    →  pred['e'] = 'ez'
   a='er'   →  pred['er'] = 'erre'
   a='era'  →  pred['era'] = 'eraman'
   ```

   eta, beraz, aurreztutako karaktere kopurua `n=len(w)-len(a)=6-3=3` litzateke.

   (c) `percent_saved_chars(pred, filename, encoding)`
   Hutsunez banandutako hitzez osotutako testu fitxategi batetako hitz guztiak idaztean, hiztegi iragarle bati esker aurrezten diren karaktereen portzentaia (ehunekotan) bueltatzen du.

2. (**3 puntu**) Demagun `n = len(z)` tamainako `z` zerrenda bat dugula, zenbaki oso ez errepikatuez osotua. Ondoko hiru funtzioek `z` zerrendan `b` balioa batzen duten bi elementu existitzen ote diren konprobatzen dute. Funtzio bakoitzaren konplexutasuna aztertu zerrendaren `n` luzerarekiko eta notazio asintotikoan adierazi.

```python
def batura_zehatza_v1(z, b):
    s = set(z)
    for x in z:
        if b-x in s and x!=b-x:
            return True
    return False


def batura_zehatza_v2(z, b):
    for i in range(len(z)-1):
        for j in range(i+1, len(z)):
            if z[i]+z[j] == b:
                return True
    return False
```

```
def batura_zehatza_v3(z, b):
    z = sorted(z)
    i = 0
    j = len(z)-1
    while i<j:
        b2 = z[i]+z[j]
        if b2==b:
            return True
        elif b2<b:
            i+=1
        else :
            j-=1
    return False
```

3. (**4 puntu**) Demagun `[0,n]` tarte batetako indize multzo bat adieraz dezakeen **`IndexSet`** klasea sortu nahi dugula. Halako objektu bat eratzeko balio boolearrez betetako `n+1` tamainako zerrenda bat erabil genezake, zeinetan `i` posizioko boolearrak indize multzoan `i` indizea ote dagoen adierazten duen. Demagun adibidez `[0,7]` tartean definitutako eta $\{0,3,4,6\}$ indizeez osotutako `IndexSet` objektu bat eraiki nahi dugula. Bere edukia adierazteko ondoko zerrenda erabili genezake: `[True, False, False, True, True, False, True, False]`.

   Sortu itzazu ondoko metodoak izango dituen `IndexSet` klasea, ahalik eta konplexutasun tenporal txikieneko kodea erabiliz:

   - `__init__(self, n, it)`: `[0,n]` tartean definitutako indize multzo hutsa sortu eta `it` iteragarriko indize guztiak gehitzen ditu. Indizeren bat `[0,n]` tartetik kanpo badago, `IndexError` motako salbuespen bat jaurtikitzen du.
   - `add(self, i)`: `i` indizea gehitzen du. Indizea `[0,n]` tartetik kanpo badago, `IndexError` motako salbuespen bat jaurtikitzen du.
   - `rank(self)`: indize multzoaren heina (`n`) bueltatzen du
   - `remove(self, i)`: `i` indizea ezabatzen du. Indizea multzoan ez badago, `IndexError` motako salbuespen bat jaurtikitzen du.
   - `__contains__(self, i)`: `i` indizea multzoan ote dagoen bueltatzen du.
   - `__len__(self)`: multzoaren tamaina (indize kopurua) bueltatzen du.
   - `__eq__(self, other)`: `self` indize multzoa `other`-en baliokidea ote den bueltatzen du. Bi indize multzo baliokideak dira baldin eta soilik baldin heina berdina badute (tarte berean definituak badaude) eta indize berdinak badituzte.
   - `__iter__(self)`: multzoko indizeen gaineko iteradore bat bueltatzen du.
   - `__str__(self)`: indize multzoaren testu errepresentazio bat bueltatzen du.
   - `__repr__(self)`: indize multzoaren testu errepresentazio kanonikoa bueltatzen du.
   - `union(self, other)`: `self` eta `other` indize multzoen bilkura bueltatzen du (indize multzo berri bat). Emaitzako indize multzoaren heina (definizio tartea) `self` eta `other` multzoko heinen arteko maximoa da.
   - `intersection(self, other)`: `self` eta `other` indize multzoen ebakidura bueltatzen du (indize multzo berri bat). Emaitzako indize multzoaren heina (definizio tartea) `self` eta `other` multzoko heinen arteko minimoa da.