resitev

January 28, 2024

1 Day 21: Allergen Assessment

(Povezava na nalogo)

Za moj okus ena lepših nalog.

Podatki so v takšni obliki:

```
mxmxvkd kfcds sqjhc nhms (contains dairy, fish)
trh fvjkl sbzzf mxmxvkd (contains dairy)
sqjhc fvjkl (contains soy)
sqjhc mxmxvkd sbzzf (contains fish)
```

Vsaka vrstica predstavlja jed. Levo so sestavine v neznanem jeziku, v oklepaju so alergeni, ki jih te sestavine vsebujejo.

- Vsaka sestavina vsebuje kvečjemu enega od alergenov.
- Vsak alergen je v natančno eni sestavini, ne večih.
- Podatki o alergenih so pomanjkljivi: lahko se zgodi, da imamo sestavino z alergenom, vendar ta ni napisan na desni.

Prvi del: Nekatere sestavine ne morejo vsebovati alergenov. Ne zanima nas, koliko jih je, temveč kakšno je njihovo skupno število pojavitev. V gornjem primeru se izkaže, da kfcds, nhms, sbzzf in trh ne morejo vsebovati alergenov; število njihovih pojavitev je 5, saj se sbzzf pojavi dvakrat.

Drugi del: Zanima nas seznam sestavin, ki vsebujejo alergene, urejen po abecedi alergenov, ki jih vsebujejo.

1.1 Branje podatkov

Vrstice bomo prebrali in razdelili na dvoje glede na `(contains`.

[1]:
```
[line.split(" (contains ") for line in open("example.txt")]
```

[1]:
```
[['mxmxvkd kfcds sqjhc nhms', 'dairy, fish)\n'],
 ['trh fvjkl sbzzf mxmxvkd', 'dairy)\n'],
 ['sqjhc fvjkl', 'soy)\n'],
 ['sqjhc mxmxvkd sbzzf', 'fish)']]
```

Za vsak par iz tega seznama storimo naslednje: prvega razdelimo z običajnim `split`, drugemo pa s `strip(")\n")` odbijemo zaklepaje in znake za novo vrstico, nato pa ga razbijemo glede na `", "`. Obe strani spravimo v množici.

[2]:
```
food = [
    (set(ingredients.split()), set(allergens.strip(")\n").split(", ")))
    for ingredients, allergens in (
        line.split(" (contains ") for line in open("example.txt")
    )
]
```

[3]:
```
food
```

[3]:
```
[({'kfcds', 'mxmxvkd', 'nhms', 'sqjhc'}, {'dairy', 'fish'}),
 ({'fvjkl', 'mxmxvkd', 'sbzzf', 'trh'}, {'dairy'}),
 ({'fvjkl', 'sqjhc'}, {'soy'}),
 ({'mxmxvkd', 'sbzzf', 'sqjhc'}, {'fish'})]
```

1.2 Možni alergeni

Osnovna opazka je tale. V prvi vrstici imamo sestavino, ki vsebuje mleko in sestavino, ki vsebuje ribe. Torej mora ena od sestavin `{'kfcds', 'mxmxvkd', 'nhms', 'sqjhc'}` vsebovati mleko (in ena od sestavin teh mora vsebovati ribe, a to zdajle ni pomembno). Iz druga vrstice razberemo, da ena od sestavin `{'fvjkl', 'mxmxvkd', 'sbzzf', 'trh'}` vsebuje mleko. **Ker je vsak alergen le v eni sestavini**, to pomeni, da mora biti mleko v eni od sestavin, ki so v preseku teh dveh množic. (To je tule le `mxmxvkd`, a načelno bi lahko bilo kandidatov tudi več.)

Potrebno je torej iti prek vseh jedi in izračunati preseke vseh množic sestavin jedi, v katerih se pojavi posamični alergen.

[4]:
```
ing_by_all = {}
for ingredients, allergens in food:
    for allergen in allergens:
        if allergen not in ing_by_all:
            ing_by_all[allergen] = ingredients.copy()
        else:
            ing_by_all[allergen] &= ingredients
```

Množica `ing_by_all` bo imela za ključe alergene, pripadajoči elementi bodo množice sestavin, ki bi utegnile vsebovati ta alergen.

Te množice so, kot smo ugotovili zgoraj, presek vseh sestavin jedi, ki vsebujejo ta alergen. Sestavimo jih tako, da takrat, ko prvič naletimo na alergen, naredimo kopijo te množice, ko na alergen naletimo ponovno, pa izračunamo presek te množice s sestavinami nove jedi s tem alergenom.

Tako dobimo naslednji slovar kandidatov.

[5]:
```
ing_by_all
```

[5]:
```
{'fish': {'mxmxvkd', 'sqjhc'}, 'dairy': {'mxmxvkd'}, 'soy': {'fvjkl', 'sqjhc'}}
```

1.3 Prirejanje alergenov

Odtod naprej gre enako kot v nalogi 16, Ticket Translation. Če je naloga enolično rešljiva (in je), potem obstaja nek alergen, za katerega že vemo, v kateri sestavini je. V gornjem primeru je to mleko, ki je v `mxmxvkd`.

Ker vsebuje vsaka sestavina le en alergen, lahko to sestavino pobrišemo iz množic kandidatov za vseh ostalih alergene. V gornjem primeru so ribe bodisi v mxmxvkd ali sqjhc. Ker vemo, da mxmxvkd vsebuje mleko, ne more vsebovati tudi rib, torej lahko po prvem koraku, spremenimo slovar kandidatov v

```
{'fish': {'sqjhc'}, 'soy': {'fvjkl', 'sqjhc'}}
```

Zdaj imamo naslednji alergen, za katerega zagotovo vemo, v kateri sestavini je vsebovan. Ponovimo vajo. In ponavljamo jo, dokler ne najdemo vseh parov.

[6]:
```
assignments = {}
while ing_by_all:
    allergen, ingredients = min(ing_by_all.items(), key=lambda x: len(x[1]))
    assert len(ingredients) == 1

    del ing_by_all[allergen]
    for candidates in ing_by_all.values():
        candidates -= ingredients

    ingredient = ingredients.pop()
    assignments[ingredient] = allergen
```

V zanki poiščemo par `allergen, ingredients`, ki vsebuje minimalno število sestavin.

Vrstica `assert` bo sprožila napako, če število sestavin ni enako 1. Če se to zgodi, vemo, da smo se zmotili pri programiranju ali pa naloga ni tako preprosta, kot smo predpostavili. (Vendar: je.)

Iz slovarja nato pobrišemo ta alergen, iz vseh množic kandidatov pa odstranimo to sestavino. Množica `ingredients` ima sicer le en element, torej bi deloval tudi `remove`. Vendar metoda `remove` preverja, ali množica v resnici vsebuje element, ki ga odstranjujemo, in javi napako, če ga ni. Pred `remove` bi zato potrebovali `if`. Odštevanje množic pa deluje tudi, če množica, od katere odštevamo, ne vsebuje vseh teh elementov, torej je za nas tule bolj praktično.

Nato iz množice poberemo to sestavino in zabeležimo, da ta sestavina vsebuje ta (in le ta) alergen.

[7]:
```
assignments
```

[7]:
```
{'mxmxvkd': 'dairy', 'sqjhc': 'fish', 'fvjkl': 'soy'}
```

1.4 Prvi del

Sestavimo množico vseh alergenih jedi: to so pač ključi slovarja `assignments`. Da preštejemo pojavitve nealergenih, gremo čez sestavine vse jedi (`for ingredients, _ in food`), za vsako izračunamo množico nealergenih sestavin (`ingredients - allergenic`) ter seštejemo velikosti teh množic.

[8]:
```
allergenic = set(assignments)
print(sum(len(ingredients - allergenic) for ingredients, _ in food))
```

```
5
```

1.5 Drugi del

Izpisati moramo ključe slovarja `assignments`, urejene po abecednem redu pripadajočih vrednosti.

Pokličemo torej `sorted(assignments)`, z argumentom `key` pa podamo funkcijo, ki bo vračala ključ, po katerem je potrebno primerjati elemente. Ta ključ pa je kar `assignments.get`, ki za vsak ključ slovarja vrne njegovo vrednost.

[9]:
```
print(",".join(sorted(assignments, key=assignments.get)))
```

```
mxmxvkd,sqjhc,fvjkl
```

1.6 Vse skupaj

[10]:
```
food = [
    (set(ingredients.split()), set(allergens.strip(")\n").split(", ")))
    for ingredients, allergens in (
        line.split(" (contains ") for line in open("input.txt")
    )
]


ing_by_all = {}
for ingredients, allergens in food:
    for allergen in allergens:
        if allergen not in ing_by_all:
            ing_by_all[allergen] = ingredients.copy()
        else:
            ing_by_all[allergen] &= ingredients


assignments = {}
while ing_by_all:
    allergen, ingredients = min(ing_by_all.items(), key=lambda x: len(x[1]))
    assert len(ingredients) == 1

    del ing_by_all[allergen]
    for candidates in ing_by_all.values():
        candidates -= ingredients

    ingredient = ingredients.pop()
    assignments[ingredient] = allergen
```

```
allergenic = set(assignments)
print(sum(len(ingredients - allergenic) for ingredients, _ in food))

print(",".join(sorted(assignments, key=assignments.get)))
```

```
2075
zfcqk,mdtvbb,ggdbl,frpvd,mgczn,zsfzq,kdqls,kktsjbh
```