1. Nevarne povezave

Oddelek za gozdarske dejavnosti in motorni promet želi z namenom povečanja varnosti prepovedati kolesarjenje na povezavah, na katerih se zgodi največ kolesarskih nesreč. Raziskava, ki so jo naročili, je pokazala, da je največ nesreč na odsekih, ki jih uporablja največ kolesarjev, zato je očitno logično prepovedati kolesarjenje po njih.

Napišite funkcijo `povezave(pot)`, ki prejme neko pot (niz z imeni križišč, med katerimi so znaki -) in vrne vse odseke na tej poti kot množico parov imen. Klic `povezave("Ana-Berta-Cilka-Berta-Dani")` vrne `{("Ana", "Berta"), ("Berta", "Cilka"), ("Cilka", "Berta"), ("Berta", "Dani")}`.

Nato napišite funkcijo `popularni(poti, k)`, ki prejme seznam poti, ki so jih opravili kolesarji in vrne `k` odsekov, ki se pojavijo na največ poteh. Če se isti odsek na neki poti pojavi večkrat, ga štejte le enkrat. Če si `k`-to mesto deli več odsekov, je vseeno, katere od izenačenih odsekov vrnete. Primer je v testih.

2. Minuta za Angelco

Angelca je odkrila, da bi se dalo v zapisu poti uporabiti različna števila minusov; ta bi pomenila čas (v minutah), ki ga je kolesar porabil za določeno povezavo.

Napišite funkcijo `casi(pot)`, ki prejme takšen opis poti in vrne slovar, katerega ključi so pari križišč, vrednosti pa število minusov med tem parom.

Klic `casi("Ana----Berta--Dani------Cilka---Berta-------Ema-Berta--Ana")` vrne `{("Ana", "Berta"): 4, ("Berta", "Dani"): 2, ("Dani", "Cilka"): 6, ("Cilka", "Berta"): 3, ("Berta", "Ema"): 7, ("Ema", "Berta"): 1, ("Berta", "Ana"): 2}`.

Predpostaviti smete, da se vsak par pojavi le enkrat.

Če je naloga pretežka, predpostavite še, da se tudi vsako križišče pojavi le enkrat, vendar dobite v tem primeru največ 75 % možnih točk naloge.

3. Povzročanje gneče, iskanje nesreče

Nekateri kolesarji delajo gnečo in izzivajo nesrečo z vožnjo v krogih, na primer `"Ana-Berta-Cilka-Ana-Berta-Cilka-Ana-Berta-Cilka-Ana-Berta-Cilka"`. Napišite funkcijo `krozenje(pot)`, ki vrne `True`, če pot predstavlja takšno kroženje in `False` sicer. Koliko križišč se ponavlja (v tem primeru so tri) in kolikokrat (tule štirikrat) ni znano.

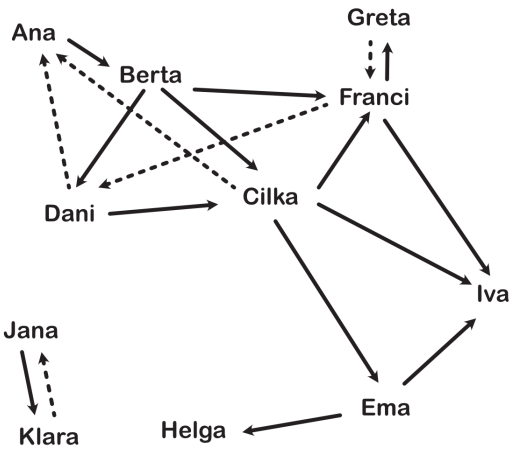
Beri naprej: pravzaprav nekateri kolesarji naredijo tako: prevozijo neko število križišč. Nato ponavljajo ta križišča, vendar v nadaljnjih krogih včasih zamenjajo eno od križišč – vedno največ eno na krog, ne pa nujno vedno isto. Primer takšne poti je `"Ana-Berta-Cilka-`**`Ana-Berta-Dani`**`-Ana-Ema-Cilka-`**`Ana-Berta-Ana-`**`Ema-Berta-Cilka"`. Prvič je zamenjal Cilko z Dani, drugič Berto s Cilko in tretjič Berto z, uh, Ano, četrtič Ano z Berto. Vaša funkcija naj vrne `True` tudi v takšnih primerih.

Za 75 % točk lahko rešite tudi preprostejšo različico, po kateri zahtevamo točne ponovitve.

Namig: preprosto poskusite vse možne dolžine krogov – 2, 3, 4, 5 ...

4. Detektiv

Kot ste uganili, so križišča poimenovana po uslužbencih OGDMP. Na vsakem je stal eden od njih in beležil promet. Rezultat tega je, da vsak od njih ve tudi, proti katerim križiščem so kolesarji vozili iz križišč, ki so jih oni dan nadzorovali.




```
povezave = {
    "Ana": {"Berta"},
    "Berta": {"Dani", "Cilka", "Franci"},
    "Cilka": {"Ema", "Franci", "Iva", "Ana"},
    "Dani": {"Cilka", "Ana"},
    "Ema": {"Iva", "Helga"},
    "Franci": {"Greta", "Iva", "Dani"},
    "Greta": {"Franci"},
    "Jana": {"Klara"},
    "Klara": {"Jana"}
}
```

pomeni, da so se od, recimo, Dani vozili proti Cilki in Ani.

Ker se je ravno tisti dan zgodil na enem od križišč tudi umor, nas zanima, ali je možno, da je kateri od kolesarjev, ki so jih videli na določenem križišču, prišel na križorišče umora.

Lažja varianta (do 75 %): napišite funkcijo `detektiv(odkod, kam, povezave)`, ki vrne `True`, če je možno, da je šel kdo od `odkod` do `kam`. Pri tem smete predpostaviti, da je zemljevid tak, da se nikoli ni možno vrniti v križišče, v katerem smo nekoč že bili (gornja slika brez črtkanih povezav). Klic `detektiv("Ana", "Iva")` vrne `True`, ker je možno, da je šel kdo od Ane do Berte, od ondod k Cilki in potem k Ivi.

Prava varianta: po zemljevidu se je možno tudi vračati v že obiskana križišča (torej obstajajo tudi črtkane povezave). Funkcija pa dobi še en argument: ko jo bodo klicali testi, bodo kot njegovo vrednost vedno podali prazno množico, vi pa počnite z njim, kar želite. (Namig: če bo funkcija rekurzivna, si boste vanjo nekaj shranjevali. Če ne bo rekurzivna, tega argumenta morda ne potrebujete in ga ignorirajte, vseeno pa ga sprejmite, da bodo testi vedeli, da rešujete to različico.)

Funkcija ne sme spreminjati podanega slovarja.

5. Strava

Napišite razred `Strava`.

- Konstruktor prejme seznam segmentov, recimo `Strava(["Ana-Berta-Dani-Ema", "Ana-Berta-Cilka", "Berta-Cilka", "Cilka-Ema-Dani-Ana-Helga", "Helga-Greta-Ema"])`,
- Metoda `dodaj(pot)`, prejme neko pot, ki jo je prevozil nek kolesar, na primer `s.dodaj("Ema-Ana-Berta-Dani-Ema")`.
- `Metoda najpopularnejsi()` vrne tistega od segmentov, podanih konstruktorju, ki se je pojavil v največ poteh, ki smo jih dodali z `dodaj(pot)`. Če se isti segment večkrat pojavi na isti poti, ga štejte le enkrat. Če si mesto najpopularnejšega deli več segmentov, lahko vrnete kateregakoli od njih.