Čakáme a čakáme

Čakanie v rade je znervózňujúce aj bez dodržiavania dvojmetrového odstupu. Je to jedna z mnoho vecí, ktoré všetci nenávidíme, no nevieme sa ich vyhnúť. Vedeli ste, že ľudia stoja v rade v priemere až 18 hodín ročne? Možno že sa vám to nezdá byť príliš veľa, no ak zoberieme priemernú dĺžku života človeka, znamená to, že v rade strávime sedem až osem týždňov nášho života. V rade čakáme naozaj všade: v obchodoch, na pošte, v jedálňach, pri nástupe do autobusu a lietadla, atď. A ani nespomenieme dilemu, ktorá nastane ak zrazu otvoria nové okienko alebo pokladňu a v priebehu jednej sekundy sa musíme rozhodnúť, či ostaneme v rade alebo sa presunieme do nového radu, kde nás čaká to isté čakanie.

Našťastie ale máme k dispozícii potenciál modernej techniky a práve preto boli vyvinuté systémy pre manažment radov (queue management system). Vy ako budúci vývojári inteligentných systémov musíte veriť v prospešný účinok aplikácie takýchto riešení, a predmetom tejto úlohy bude implementovať jeden takýto systém. Pre jednoduchosť riešenia aplikačnú doménu zúžime na supermarkety kde máme k dispozícii všetky informácie potrebné pre úspešný manažment radov.

Vstupom riešenia bude zoznam všetkých radov – otvorených pokladní – spolu s informáciami o pokladníkovi a o ľuďoch čakajúcich v rade a o ich nákupe. Každý pokladník je prezentovaný dvomi údajmi: meno a počet produktov, ktoré v priemere naskenuje za sekundu. Ku každej pokladni prislúcha aj zoznam zákazníkov s tromi údajmi: koľko produktov nakúpili, ako rýchlo si dokážu odkladať nákup do tašky a akým spôsobom chcú platiť. Ako ukážkový príklad zoberieme obchod s tromi kasami (rovnaký príklad nájdete v súbore `sample1.txt`):

> Daniel, 2 – 20, slow, card; 3, quick, card; 30, slow, cash; 17, normal, card
> Katka, 1.5 – 13, slow, cash; 24, normal, card;
> Robo, 1 – 5, normal, cash; 11, quick, card; 8, slow, card
> Sandra, 1.25 – 38, slow, cash

Prvú pokladňu operuje Daniel, ktorý je najrýchlejší zo všetkých a dokáže naskenovať dva produkty za sekundu. V jeho rade stoja ž štyria zákazníci: prvý z nich nakúpil 20 položiek, je pomalý pri ukladaní nákupu a chce platiť kartou. Druhý zákazník kúpil iba 3 veci, dokáže ich rýchlo odložiť, a bude platiť tiež kartou. Tretí zákazník si kúpil toho viac, 30 položiek, je pomalý pri ukladaní nákupu, a čo je ešte horšie, bude platiť hotovosťou. Posledný zákazník má v košíku 17 produktov, má priemernú rýchlosť a bude platiť kartou. Informácie o ďalších radoch sú definované obdobne.

Pri výpočte čakacej doby budeme vychádzať z niekoľkých predpokladov. Prvý z nich je, že zákazníci stihnú všetko vykladať na pokladničný pás ešte pred tým, než odíde človek stojaci v rade pred nimi. To znamená, že čakaciu dobu celého radu vieme vypočítať ako súčet dĺžky interakcie s jednotlivými zákazníkmi. Pritom každá interakcia sa skladá z troch častí: skenovanie, balenie a platba.

Dĺžka skenovania závisí od rýchlosti pokladníka a počtu položiek. Vypočítate ju ako podiel počtu položiek a priemernej rýchlosti skenovania, pričom vždy zaokrúhľujete smerom hore, teda rátate každú začatú sekundu. V prípade Daniela a jeho zákazníkov skenovanie potrvá 10 sekúnd (20 / 2

= 10), 2 sekundy (3 / 2 = 1,5 a zaokrúhľujeme hore), 15 sekúnd (30 / 2 = 15) a 9 sekúnd (17 / 2 = 8,5 a zaokrúhľujeme hore).

Dĺžka balenia závisí od rýchlosti zákazníka a počtu položiek, a vypočítate ju obdobne ako skenovanie, rozdiel je ale, že rýchlosť zákazníka je definovaná ako konštanta podľa kategórií: rýchly zákazník si odloží 6 položiek za sekundu, priemerný 4 a pomalý iba 3. Ako aj v prípade skenovania, musíte rátať každú začatú sekundu. V prípade prvého radu teda balenie potrvá 7 sekúnd (20 / 3 = 6,67 a zaokrúhľujeme hore), 1 sekundu (3 / 6 = 0,33 a zaokrúhľujeme hore), 10 sekúnd (30 / 3 = 10) a 5 sekúnd (17 / 4 = 4,25 a zaokrúhľujeme hore) pre jednotlivých zákazníkov.

Dĺžku platby vieme vypočítať priamočiarejšie podľa niekoľkých pravidiel:
- platba kartou potrvá vždy 5 sekúnd;
- platba hotovosťou pre rýchleho zákazníka trvá 10 sekúnd;
- platba hotovosťou pre priemerného zákazníka trvá 15 sekúnd;
- platba hotovosťou pre pomalého zákazníka trvá 25 sekúnd.

Ak teda zoberieme všetko do úvahy, v prvom rade by sme čakali 99 sekúnd, pričom:
- na obsluhu prvého zákazníka potrebujeme 10 + 7 + 5 = 22 sekúnd
- na obsluhu druhého zákazníka potrebujeme 2 + 1 + 5 = 8 sekúnd
- na obsluhu tretieho zákazníka potrebujeme 15 + 10 + 25 = 50 sekúnd
- a na obsluhu štvrtého zákazníka potrebujeme 9 + 5 + 5 = 19 sekúnd.

Na základe týchto poznatkov potom vieme implementovať náš jednoduchý systém.

Úloha 1 – 1 bod

Implementujte funkciu `load_queue_data`, ktorá má jeden parameter a to cestu k súboru, ktorý obsahuje informácie o radoch čakajúcich pri jednotlivých pokladniach v obchode. V súbore sú dva typy riadkov:
- prvý typ obsahuje informácie o pokladníkovi, obsahuje dve hodnoty oddelené čiarkou: meno a rýchlosť pokladníka
- druhý typ obsahuje informácie o zákazníkovi, obsahuje tri hodnoty a to počet nakúpených položiek, rýchlosť zákazníka pri balení a formu platby.

Príklady na takéto súbory nájdete v priečinku `samples`.

Funkcia `load_queue_data` vracia jednu hodnotu, a to načítané údaje v slovníku. Kľúčmi tohto slovníka sú pritom dvojice s menom a rýchlosťou pokladníka a hodnoty sú zoznamy trojíc, kde každá trojica definuje jedného zákazníka. Príklady správne načítaných knižníc nájdete v súboru `problem1_sample_tests.py`.

Pozor! Pri hodnotení vášho riešenia sa bude kontrolovať aj schopnosť funkcie `load_queue_data` spracovať aj nesprávne štruktúrované súbory – v tomto prípade musí vyhodiť chybu `RuntimeError`, ktorá upozorní používateľa na nesprávnosť údajov v súbore. Takisto si dávajte pozor na správne typy načítaných údajov vo výslednom slovníku.

Úloha 2 – 0,5 boda

Implementujte funkciu `calculate_customer_time`, ktorá vypočíta čas potrebný na obsluhu konkrétneho zákazníka a vracia ju v sekundách ako celé číslo. Funkcia má dva parametre: rýchlosť pokladníka a n-ticu hodnôt popisujúcu zákazníka (n-tica je jeden prvok zo zoznamu zákazníkov v rade, teda trojica hodnôt počet položiek, rýchlosť zákazníka, a forma platby).

Napríklad pri volaní funkcie s hodnotami zodpovedajúcej prvému zákazníkovi v rade Daniela funkcia vracia hodnotu 22: `calculate_customer_time(2, (20, 'slow', 'card'))`.

Úloha 3 – 0,5 boda

Samozrejme vývojári nie sú úplne nesebeckí a chcú aby aj oni mali úžitok zo svojich riešení. Implementujte funkciu `choose_queue`, ktorá pomôže vám alebo ľubovoľnému používateľovi nájsť rad, ku ktorému by sa mal pridať aby čakal čo najmenej. Funkcia má jeden parameter, a to slovník popisujúci jednotlivé rady. Tento slovník má štruktúru zodpovedajúcu výstupu funkcie `load_queue_data`. Funkcia vracia jednu hodnotu, a to meno pokladníka, do radu ktorého sa máme postaviť.

Pre náš príklad volanie funkcie s načítanými údajmi by vrátilo meno Roba. Aj keď on je najpomalší pokladník, v jeho rade postojíme iba 56 sekúnd (v ostatných by to bolo: Daniel – 99 s, Katka – 66 s, Sandra – 69 s).

Úloha 4 – 1 bod

Ako veľmi často vo svete, aj pri radoch majú rôzni aktéri rôzne ciele. Pričom zákazník hľadá ten rad, v ktorom bude čakať najmenej, pre optimálne fungovanie obchodu cieľom je znížiť čo najviac celkovú dobu čakania. Keďže ale pokladne fungujú súbežne, nás zaujíma najdlhší čas potrebný pre obsluhu jedného radu. Ak zoberieme náš príklad, tak jednotliví pokladníci potrebujú 99, 66, 56 a 69 sekúnd pre obsluhu svojich radov. Pre vás ako manažéra obchodu je ale dôležitý iba údaj 99 sekúnd, pretože presne toľko potrvá, kým odíde posledný zákazník.

Implementujte funkciu `where_to_send`, ktorá vracia meno pokladníka radu, do ktorého potrebujete nasmerovať nového zákazníka tak, aby chod obchodu bol optimálny. Parametrami funkcie sú informácie o súčasnom stave radov (slovník) a n-tica popisujúca nového zákazníka.

Napríklad pri volaní funkcie `where_to_send(sample_row, (58, 'slow', 'card'))` sa vráti hodnota Katka, pretože všetkých zákazníkov obslúžime najrýchlejšie práve vtedy, ak nový zákazník sa postaví do jej radu.

Úloha 5 – 1 bod

Asi najväčšia dilema manažéra prevádzky je otázka, či sa má otvoriť nová pokladňa. Síce takto vieme obslúžiť viac zákazníkov v menšom čase, možno že by sme tak ušetrili len pár sekúnd, a ďalšiemu pokladníkovi by sme mohli dať aj viac užitočnú úlohu.

Implementujte funkciu `should_use_cashier`, ktorá odpovie práve na túto otázku. Funkcia má tri parametre: informácie o voľnom pokladníkovi (n-tica typu kľúča slovníka, teda meno a

rýchlosť), informácie o súčasnom stave radov (slovník ako z `load_queue_data`), a nepovinný parameter `save_at_least` s defaultnou hodnotou 30. Posledný parameter určuje minimálny počet ušetrených sekúnd pri ktorom sa nám oplatí otvoriť novú pokladňu. Funkcia vracia jednu booleovskú hodnotu: `True`, ak pokladňu máme otvoriť, `False` v opačnom prípade.

Pri otvorení novej pokladne budeme predpokladať, že k nej prídu poslední členovia ostatných radov, ak nie sú práve na rade (pred nimi stojí aspoň jedna osoba).

Ak máme k dispozícii voľného pokladníka Mareka s rýchlosťou štyri, ušetrili by sme 19 sekúnd. Teda volania funkcie by vrátili nasledovné hodnoty:

```
should_use_cashier(('Marek', 4), queues) -> False
should_use_cashier(('Marek', 4), queues, save_at_least=10) -> True
```

Kostra riešenia obsahuje ešte prázdnu hlavnú funkciu `main`, ktorú môžete využiť na testovanie. Pri riešení môžete vytvoriť ľubovoľné pomocné funkcie a môžete použiť hotové riešenia z ľubovoľného štandardného modulu jazyka Python.

Vaše riešenia môžete otestovať aj pomocou sady testov v súbore `problem1_sample_tests.py`. Pri hodnotení vášho riešenia použijeme podobné testy, avšak ich bude viac.