PHARMDB – system zarządzania i analizy leków

Napisz program służący do zarządzania kompleksową bazą danych leków oraz analizy ich wzajemnych zależności. System ma zapewniać efektywne odpowiedzi na zapytania dotyczące bezpieczeństwa przyjmowanych leków i optymalizacji ich doboru. System ma na celu wspomaganie lekarzy i farmaceutów w podejmowaniu decyzji terapeutycznych poprzez analizę działań niepożądanych, wskazywanie najlepszych leków dla jakieś choroby oraz sugerowanie alternatywnych opcji leczenia. System musi uwzględniać, że informacje o lekach mogą być dynamicznie aktualizowane – mogą pojawiać się nowe działania niepożądane, nowe zamienniki lub zmiany we wskazaniach terapeutycznych. Te aktualizacje powinny być natychmiast uwzględniane w analizach i rekomendacjach generowanych przez system.

Struktura informacji o lekach

Każdy lek powinien być opisany następującymi atrybutami:

- `id` – unikalny identyfikator nadawany przez system zgodnie z kolejnością, np. "D0001", "D0002".
- `name` – unikalna nazwa leku, np. "Apap", "Ibuprom".
- `indications` – lista wskazań terapeutycznych, tzn. chorób lub objawów na które działa dany lek, np. "nadciśnienie tętnicze", "ból głowy", wraz z poziomem skuteczności leku dla danego wskazania (skala 1-10).
- `substitutes` – lista leków, które mogą być zastąpione przez dany lek. *Uwaga: relacja ta nie jest symetryczna ani przechodnia. Proszę również zwrócić uwagę na dokładny kierunek zastępowalności.* Przykład: jeśli lek A ma na liście substitutes lek B, oznacza to, że lek B może być zastąpiony przez lek A lub inaczej A może zastąpić B, ale nie odwrotnie.
- `side_effects` – lista działań niepożądanych, gdzie każde działanie ma:
  - nazwę objawu, np. "zawroty głowy", "nudności";
  - poziom dolegliwości: łagodny (`MINOR`, wartość 1), istotny (`SIGNIFICANT`, wartość 2) lub krytyczny (`CRITICAL`, wartość 3);
  - częstotliwość występowania w procentach, np. 15.5 oznacza, że występuje średnio u 15.5% pacjentów.

Wymagania

Należy napisać klasę `PharmDB` wraz z odpowiednimi atrybutami i następującymi metodami:

1. `__init__(self)` – inicjalizacja bazy danych leków oraz odpowiednich struktur danych.
2. `add_drug(self, drug_name, indications=None, substitutes=None, side_effects=None)` – dodanie nowego leku do bazy danych o podanych wskazaniach terapeutycznych, lekach zastępowalnych i efektach ubocznych. System ma automatycznie przydzielić identyfikator i go zwrócić. Dodany lek może być zastępnikiem tylko dla leków wcześniej dodanych do bazy danych. Należy założyć, że lek jest dodawany do bazy danych tylko raz oraz nie ma duplikatów.

3. `number_of_indications(self, drug_id, min_efficacy)` – zwrócenie liczby wskazań terapeutycznych o efektywności co najmniej `min_efficacy` dla podanego leku.
4. `number_of_alternative_drugs(self, drug_id)` – zwrócenie liczby leków, które mogą bezpośrednio zastąpić dany lek. *Uwaga:* nie chodzi o liczbę leków, które mogą być bezpośrednio zastąpione przez dany lek.
5. `worst_side_effect(self, drug_id)` – zwrócenie nazwę nabardziej dotkliwego skutku ubocznego (skutku o największej częstotliwości spośród tych o największym poziomie dolegliwości) dla podanego leku.
6. `risk_score(self, drug_id)` – zwrócenie wskaźnika ryzyka (*risk score*) zdefiniowanego jako ważona suma wszystkich działań niepożądanych podanego leku (częstość występowania $\times$ poziom dolegliwości).
7. `find_best_alternative(self, drug_id, max_steps=2)` – zwrócenie identyfikatora leku o minimalnym ryzyku spośród leków, które można zastosować zamiast leku `drug_id` przy ograniczeniu liczby zamian. Można wykonać co najwyżej `max_steps` zamian, np. przy `max_steps=2` można używać zamienników dla bezpośrednich zamienników podanego leku. Przykład: jeśli lek `A` można zastąpić lekiem `B`, a lek `B` lekiem `C`, to wynikiem działania tej funkcji dla leku `A`, powinien być identyfikator leku o minimalnym ryzyku spośród leków `A`, `B` i `C`. *Uwaga:* W przypadku równego ryzyka należy zwrócić lek o wcześniejszym identyfikatorze.
8. `longest_alternative_list(self)` – zwrócenie listy idetyfikatorów leków stanowiącej najdłuższy ciąg zamienników leków, gdzie każdy kolejny lek na liście może bezpośrednio zamienić lek poprzedni; w przypadku list równej długości, należy zwrócić listę o wcześniejszych identyfikatorach (leksykograficznie). Przykład: jeśli A może być zastąpione przez B, B przez C lub D, a C przez D, to wynikiem działania tej funkcji powinna być lista `[id(A), id(B), id(C), id(D)]`.
9. `find_best_drug_for_indication(self, disease_name)` – zwrócenie identyfikatora leku o największej efektywności dla wskazanej choroby. W przypadku wielu leków o takiej samej efektywności należy zwrócić identyfikator najpóźniej dodanego do bazy danych leku.
10. `update_best_indication(self, disease_name, new_efficacy)` – zmiana efektywność najlepszego leku dla wskazanej choroby (jak wyżej). Funkcja może być przydatna w sytuacji gdy przy częstym stosowaniu jakiegoś leku można uaktualnić dane o jego efektywności.

Pełną specyfikację operacji i przykłady można znaleźć w pliku `pharmdb.py`.

Złożoność

Niech D i S oznaczają odpowiednio liczbę leków oraz sumaryczną liczbę zamienników w bazie danych, a K maksymalną liczbę leków dla dowolnego wskazania terapeutycznego.

Wymagania czasowe na maksymalną liczbę punktów:

- funkcje `__init__`, `number_of_indications`, `number_of_alternative_drugs`, `worst_side_effect`, `risk_score` oraz `find_best_drug_for_indication`

powinny działać w czasie $O(1)$;
- funkcja `add_drug` powinna działać w czasie $O(k \log K + s + e)$, gdzie k, s i e są odpowiednio liczbą wskazań terapeutycznych (`len(indications)`), liczbą zamienników (`len(substitutes)`) oraz liczbą działań niepożądanych (`len(side_effects)`) dla dodawanego leku;
- funkcja `find_best_alternative` w praktyce powinna działać istotnie szybciej niż $O(D+S)$, a ostateczna ocena będzie zależeć od uzyskanej złożoności czasowej (w szczególności funkcja nie powinna przeglądać wszystkich leków jeśli nie ma takiej potrzeby);
- funkcja `longest_alternative_list` powinna działać w czasie $O(d)$, gdzie d to długość zwracanej listy identyfikatorów leków;
- funkcja `update_best_indication` powinna działać w czasie $O(\log K)$.

Można przyjąć, że podstawowe operacje (sprawdzenie obecności, dodanie, odczytanie, modyfikacja i usunięcie elementu) na słownikach (`dict`) i zbiorach (`set`) działają w czasie $O(1)$. Przy korzystaniu z list należy przyjąć, że w czasie stałym wykonywane jest odczytywanie i modyfikowanie pojedynczych elementów, a także dodawanie i usuwanie elementów na końcu listy, natomiast łączenie list oraz wstawianie i usuwanie elementów ze środka bądź początku listy wymaga czasu liniowego.

Punktacja

Za cały projekt można dostać maksymalnie 30 punktów. Orientacyjna punktacja za poszczególne funkcje jest następująca:
- `klasa PharmDB` i `__init__` – 1 punkt;
- `add_drug` – 3 punkty, przy czym prawidłowa obsługa struktur danych potrzebnych dla innych funkcji będzie częściowo oceniona wraz z nimi;
- `number_of_indications`, `number_of_alternative_drugs`, `worst_side_effect`, `risk_score` – każda funkcja 1.5 punktu;
- `find_best_alternative` – 7 punktów (będzie zależeć od uzyskanej złożoności czasowej),
- `longest_alternative_list` – 7 punktów;
- `find_best_drug_for_indication` i `update_best_indication` – każda funkcja 3 punkty;
- gorsza złożoność czasowa będzie skutkować utratą punktów;
- brak opisu struktury danych, metod i zmiennych może skutkować utratą do 5 punktów.

Aby projekt podlegał ocenie, przesłane rozwiązanie musi zawierać implementację minimalnej funkcjonalność w postaci klasy `PharmDB` oraz metod `__init__`, `add_drug`, `number_of_indications`, `number_of_alternative_drugs`, `risk_score` i `find_best_alternative`.

Można korzystać z kodów z wykładów i laboratoriów. Rozwiązanie należy przesyłać przez Moodle'a i powinno się składać z jednego pliku `pharmdb.py`.