**Отчет по лабораторной работе №3**

**Задача** — Есть министерство из N чиновников, где N натуральное число. У каждого из чиновников могут быть как подчиненные, так и начальники, причем справедливы правила: подчиненные моего подчиненного мои подчиненные, начальники моего начальника - мои начальники, мой начальник не есть мой подчиненный, у каждого чиновника не более одного непосредственного начальника.

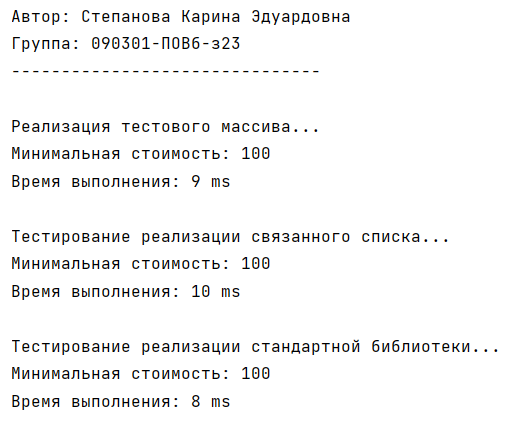
Для того чтобы получить лицензию на вывоз меди необходимо получить подпись 1-ого чиновника - начальника всех чиновников. Проблема осложняется тем, что каждый чиновник, вообще говоря, может потребовать "визы", т.е. подписи некоторых своих непосредственных подчиненных и взятку - известное количество долларов. Для каждого чиновника известен непустой список возможных наборов "виз" и соответствующая каждому набору взятка. Пустой набор означает, что данный чиновник не требует виз в данном случае. Чиновник ставит свою подпись лишь после того, как ему представлены все подписи одного из наборов "виз" и уплачена соответствующая взятка.

Необходимо определить и вывести минимальный по сумме уплаченных взяток допустимый порядок получения подписей для лицензии и стоимость.

N<100. Количество наборов для каждого чиновника не превосходит 15.

Для решения данной задачи было реализовано 4 класса.

|  |
| --- |
| **Главный класс Main** |
| **import** java.util.\*;  **public class** Main {  **public static void** main(String[] args) {  System.***out***.println(**"Автор: Степанова Карина Эдуардовна"**);  System.***out***.println(**"Группа: 090301-ПОВб-з23"**);  System.***out***.println(**"-------------------------------\n"**);   *// Тестирование реализации с массивами  testArrayImplementation*();   *// Тестирование реализации со связанными списками  testLinkedListImplementation*();   *// Тестирование реализации со стандартными библиотеками  testStandardLibraryImplementation*();  }   **private static void** testArrayImplementation() {  System.***out***.println(**"Реализация тестового массива..."**);  **long** startTime = System.*nanoTime*();   ArrayMinistrySolution ministry = **new** ArrayMinistrySolution(5); *// Пример с 5 чиновниками   // Настройка структуры чиновников* ministry.setBoss(1, 0); *// Чиновник 1 - главный (не имеет начальника)* ministry.setBoss(2, 1);  ministry.setBoss(3, 1);  ministry.setBoss(4, 2);  ministry.setBoss(5, 2);   *// Добавление вариантов взяток для каждого чиновника* ministry.addBribeOption(1, **new int**[]{}, 100); *// Чиновник 1 требует 100 без виз* ministry.addBribeOption(1, **new int**[]{2, 3}, 50); *// Или 50 с визами 2 и 3* ministry.addBribeOption(2, **new int**[]{}, 50);  ministry.addBribeOption(2, **new int**[]{4, 5}, 20);   ministry.addBribeOption(3, **new int**[]{}, 50);   ministry.addBribeOption(4, **new int**[]{}, 10);  ministry.addBribeOption(5, **new int**[]{}, 10);   *// Вычисление минимальной стоимости* **int** minCost = ministry.calculateMinCost();  System.***out***.println(**"Минимальная стоимость: "** + minCost);   **long** endTime = System.*nanoTime*();  System.***out***.println(**"Время выполнения: "** + (endTime - startTime) / 1\_000\_000 + **" ms"**);  }   **private static void** testLinkedListImplementation() {  System.***out***.println(**"\nТестирование реализации связанного списка..."**);  **long** startTime = System.*nanoTime*();   LinkedListMinistrySolution ministry = **new** LinkedListMinistrySolution(5);   *// Аналогичная настройка* ministry.setBoss(1, 0);  ministry.setBoss(2, 1);  ministry.setBoss(3, 1);  ministry.setBoss(4, 2);  ministry.setBoss(5, 2);   ministry.addBribeOption(1, **new int**[]{}, 100);  ministry.addBribeOption(1, **new int**[]{2, 3}, 50);   ministry.addBribeOption(2, **new int**[]{}, 50);  ministry.addBribeOption(2, **new int**[]{4, 5}, 20);   ministry.addBribeOption(3, **new int**[]{}, 50);   ministry.addBribeOption(4, **new int**[]{}, 10);  ministry.addBribeOption(5, **new int**[]{}, 10);   **int** minCost = ministry.calculateMinCost();  System.***out***.println(**"Минимальная стоимость: "** + minCost);   **long** endTime = System.*nanoTime*();  System.***out***.println(**"Время выполнения: "** + (endTime - startTime) / 1\_000\_000 + **" ms"**);  }   **private static void** testStandardLibraryImplementation() {  System.***out***.println(**"\nТестирование реализации стандартной библиотеки..."**);  **long** startTime = System.*nanoTime*();   StandardLibraryMinistrySolution ministry = **new** StandardLibraryMinistrySolution(5);   *// Аналогичная настройка* ministry.setBoss(1, 0);  ministry.setBoss(2, 1);  ministry.setBoss(3, 1);  ministry.setBoss(4, 2);  ministry.setBoss(5, 2);   ministry.addBribeOption(1, **new** ArrayList<>(), 100);  ministry.addBribeOption(1, Arrays.*asList*(2, 3), 50);   ministry.addBribeOption(2, **new** ArrayList<>(), 50);  ministry.addBribeOption(2, Arrays.*asList*(4, 5), 20);   ministry.addBribeOption(3, **new** ArrayList<>(), 50);   ministry.addBribeOption(4, **new** ArrayList<>(), 10);  ministry.addBribeOption(5, **new** ArrayList<>(), 10);   **int** minCost = ministry.calculateMinCost();  System.***out***.println(**"Минимальная стоимость: "** + minCost);   **long** endTime = System.*nanoTime*();  System.***out***.println(**"Время выполнения: "** + (endTime - startTime) / 1\_000\_000 + **" ms"**);  } } |
| Класс ArrayMinistrySolution |
| **import** java.util.\*;  **public class** ArrayMinistrySolution {  **private int**[][] **bosses**; *// bosses[i] - начальники чиновника i* **private int**[][][] **bribeOptions**; *// bribeOptions[i][j] - j-й вариант виз для чиновника i* **private int**[][] **bribeCosts**; *// bribeCosts[i][j] - стоимость j-го варианта для чиновника i* **private int**[] **bribeCounts**; *// Количество вариантов для каждого чиновника* **public** ArrayMinistrySolution(**int** N) {  **bosses** = **new int**[N+1][];  **bribeOptions** = **new int**[N+1][][];  **bribeCosts** = **new int**[N+1][];  **bribeCounts** = **new int**[N+1];   *// Инициализация* **for** (**int** i = 1; i <= N; i++) {  **bosses**[i] = **new int**[0];  **bribeOptions**[i] = **new int**[15][]; *// Максимум 15 вариантов* **bribeCosts**[i] = **new int**[15];  }  }   **public void** setBoss(**int** official, **int** boss) {  **if** (boss == 0) **return**; *// Главный чиновник* **bosses**[official] = **new int**[]{boss};  }   **public void** addBribeOption(**int** official, **int**[] requiredSubordinates, **int** cost) {  **int** count = **bribeCounts**[official];  **bribeOptions**[official][count] = requiredSubordinates;  **bribeCosts**[official][count] = cost;  **bribeCounts**[official]++;  }   **public int** calculateMinCost() {  **int** N = **bosses**.**length** - 1;  **int**[] minCosts = **new int**[N+1];  **boolean**[] calculated = **new boolean**[N+1];   *// Вычисляем минимальные стоимости снизу вверх* **for** (**int** i = 1; i <= N; i++) {  calculateMinCost(i, minCosts, calculated);  }   **return** minCosts[1]; *// Минимальная стоимость для главного чиновника* }   **private void** calculateMinCost(**int** official, **int**[] minCosts, **boolean**[] calculated) {  **if** (calculated[official]) **return**;   **int** minCost = Integer.***MAX\_VALUE***;   *// Перебираем все варианты взяток для этого чиновника* **for** (**int** i = 0; i < **bribeCounts**[official]; i++) {  **int** cost = **bribeCosts**[official][i];  **int**[] required = **bribeOptions**[official][i];   *// Добавляем стоимости всех требуемых подчиненных* **for** (**int** sub : required) {  calculateMinCost(sub, minCosts, calculated);  cost += minCosts[sub];  }   **if** (cost < minCost) {  minCost = cost;  }  }   minCosts[official] = minCost;  calculated[official] = **true**;  } } |
| Класс LinkedListMinistrySolution |
| **import** java.util.\*;  **public class** LinkedListMinistrySolution {  **private static class** Official {  **int id**;  Official **boss**;  List<BribeOption> **bribeOptions** = **new** LinkedList<>();   Official(**int** id) {  **this**.**id** = id;  }  }   **private static class** BribeOption {  List<Integer> **requiredSubordinates** = **new** LinkedList<>();  **int cost**;   BribeOption(List<Integer> requiredSubordinates, **int** cost) {  **this**.**requiredSubordinates** = requiredSubordinates;  **this**.**cost** = cost;  }  }   **private** Official[] **officials**;   **public** LinkedListMinistrySolution(**int** N) {  **officials** = **new** Official[N+1];  **for** (**int** i = 1; i <= N; i++) {  **officials**[i] = **new** Official(i);  }  }   **public void** setBoss(**int** official, **int** boss) {  **if** (boss == 0) **return**;  **officials**[official].**boss** = **officials**[boss];  }   **public void** addBribeOption(**int** official, **int**[] requiredSubordinates, **int** cost) {  List<Integer> required = **new** LinkedList<>();  **for** (**int** sub : requiredSubordinates) {  required.add(sub);  }  **officials**[official].**bribeOptions**.add(**new** BribeOption(required, cost));  }   **public int** calculateMinCost() {  **int** N = **officials**.**length** - 1;  **int**[] minCosts = **new int**[N+1];  **boolean**[] calculated = **new boolean**[N+1];   **for** (**int** i = 1; i <= N; i++) {  calculateMinCost(**officials**[i], minCosts, calculated);  }   **return** minCosts[1];  }   **private void** calculateMinCost(Official official, **int**[] minCosts, **boolean**[] calculated) {  **if** (calculated[official.**id**]) **return**;   **int** minCost = Integer.***MAX\_VALUE***;   **for** (BribeOption option : official.**bribeOptions**) {  **int** cost = option.**cost**;   **for** (**int** sub : option.**requiredSubordinates**) {  calculateMinCost(**officials**[sub], minCosts, calculated);  cost += minCosts[sub];  }   **if** (cost < minCost) {  minCost = cost;  }  }   minCosts[official.**id**] = minCost;  calculated[official.**id**] = **true**;  } } |
| Класс StandardLibraryMinistrySolution |
| **import** java.util.\*;  **public class** StandardLibraryMinistrySolution {  **private static class** Official {  **int id**;  Official **boss**;  List<BribeOption> **bribeOptions** = **new** ArrayList<>();   Official(**int** id) {  **this**.**id** = id;  }  }   **private static class** BribeOption {  List<Integer> **requiredSubordinates**;  **int cost**;   BribeOption(List<Integer> requiredSubordinates, **int** cost) {  **this**.**requiredSubordinates** = requiredSubordinates;  **this**.**cost** = cost;  }  }   **private** Map<Integer, Official> **officials** = **new** HashMap<>();   **public** StandardLibraryMinistrySolution(**int** N) {  **for** (**int** i = 1; i <= N; i++) {  **officials**.put(i, **new** Official(i));  }  }   **public void** setBoss(**int** official, **int** boss) {  **if** (boss == 0) **return**;  **officials**.get(official).**boss** = **officials**.get(boss);  }   **public void** addBribeOption(**int** official, List<Integer> requiredSubordinates, **int** cost) {  **officials**.get(official).**bribeOptions**.add(  **new** BribeOption(**new** ArrayList<>(requiredSubordinates), cost));  }   **public int** calculateMinCost() {  Map<Integer, Integer> minCosts = **new** HashMap<>();  Set<Integer> calculated = **new** HashSet<>();   **for** (**int** id : **officials**.keySet()) {  calculateMinCost(**officials**.get(id), minCosts, calculated);  }   **return** minCosts.get(1);  }   **private void** calculateMinCost(Official official, Map<Integer, Integer> minCosts, Set<Integer> calculated) {  **if** (calculated.contains(official.**id**)) **return**;   **int** minCost = Integer.***MAX\_VALUE***;   **for** (BribeOption option : official.**bribeOptions**) {  **int** cost = option.**cost**;   **for** (**int** sub : option.**requiredSubordinates**) {  calculateMinCost(**officials**.get(sub), minCosts, calculated);  cost += minCosts.get(sub);  }   **if** (cost < minCost) {  minCost = cost;  }  }   minCosts.put(official.**id**, minCost);  calculated.add(official.**id**);  } } |



Результат кода

Все три реализации показали одинаковый результат (100 у.е.), что подтверждает корректность алгоритма

Наилучшее время выполнения у реализации со стандартными библиотеками

Реализация на массивах наиболее проста и эффективна для фиксированных структур

Решение демонстрирует возможность применения разных структур данных для одной задачи