КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

**Семестровая работа №1**

**По алгоритмам и структурам данных**

Вариант №5

“Полином от трех переменных”

Выполнил:

Дубровец Виталий

Студент 1 курса

Группа 11-906

Проверил:

Старший преподаватель

Кафедры программной инженерии

Хадиев К.Р.

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| **Тема** | **Страница** |
| Постановка задачи | 3 |
| PolinomItem, PolinomListItem | 4 |
| PolinomList | 5 |
| Polinom3 | 6 |
| Метод toString | 8 |
| Тесты | 9 |

**Постановка задачи**

**ВАРИАНТ 5**

Класс Polinom3 - полином от трех переменных с целочисленными коэффициентами.

Представить его в виде списка, каждый элемент которого содержит значение коэффициента и значение показателей степеней переменных при этом коэффициенте.

Наборы показателей степеней должны быть всегда упорядочены в лексикографическом порядке по трем степеням соответствующих переменных (операции не должны нарушать упорядоченность).

Реализовать методы:

* **конструктор Polinom3(String filename)**: Построение списка по полиному, заданному списком коэффициентов и степеней в некотором текстовом файле;
* **String toString()**: Строковое представление полинома.
* **void insert(int coef, int deg1, int deg2, int deg3)**: Вставка элемента coef\*x^deg1 \* y^deg2 \* z^deg3 в список, не нарушая лексикографической упорядоченности. При вставке учесть существует ли подобный элемент в списке, если да, то заменить.
* **void delete(int deg1, int deg2, int deg3)**: удалить элемент, соответствующий данным показателям степени;
* **void add(Polinom3 p)**: прибавить к одному полиному другой. Использовать метод слияния.
* **void derivate(int i)**: взять частную производную полинома по i-й переменной.
* **int value(int x, int y, int z)**: вычислить значение полинома в точке (x,y,z)

**PolinomItem** - часть многочлена, содержащая название переменной и степень при данной переменной (x^3, y^2, z^0 и т.д.)  
  
**-get/set методы для степеней переменных**  
**-equals**  
**-toString**

Сложности всех методов класса

T = O(1), M = O(1)

Так как методы работают с множеством из двух элементов (название переменной и степень)  
  
**PolinomListItem** - многочлен (элемент списка многочленов), состоящий из 3-х PolinomItem`ов, коэффициента (3 \* x^2 \* y^1, -1 \* x^3 и т.д.) и ссылки на следующий PolinomListItem; Реализует Comparable для соблюдения лексикографического порядка в полиноме.  
  
**-get/set методы для степеней переменных, коэффициента и ссылки на следующий элемент списка**  
**-compareTo**  
**-equals**  
**-toString**  
**-copy** - возвращает копию объекта с теми же атрибутами и ссылкой на следующий элемент, равной null.

Сложности всех методов класса

T = O(1), M = O(1)

Так как методы работают с множеством из двух элементов (название переменной и степень)

**PolinomList** - список многочленов, состоящий из PolinomListItem. Содержит ссылку на голову списка и размер списка

Имеет конструктор, формирующий список по массиву PolinomListItem; Сложность по времени T = O(N) т.к для каждого элемента множества из N элементов производит добавление в список. M = O(1)  
  
**Методы :**  
**-add(PolinomListItem),**

**-add(int coef, int deg1, int deg2, int deg3)**  
Добавляет новый многочлен с список или заменяет уже существующий. Управляет размером списка. Для поиска подходящей позиции используются линейный поиск и метод compareTo у PolinomListItem.

Сложность по времени: T = O(N), тк как метод работает с множеством размера N и в худшем случае необходимо сделать N итераций для добавления элемента. Сложность по памяти - M =O(1).  
  
**-delete(PolinomListItem),**

**-delete(int coef, int deg1, int deg2, int deg3)**  
Находит и удаляет многочлен с соответствующими степенями, если такого нет - завершает работу метода. Управляет размером списка. Использует линейный поиск.  
T = O(N), тк как метод работает с множеством размера N и в худшем случае необходимо сделать N итераций для удаления элемента  
M = O(1)  
  
**-equals(Object o)**  
Сравнивает 2 объекта. Использует линейный поиск для сравнения соответствующих элементов.  
T = O(N), тк как метод работает с множеством размера N и в худшем случае необходимо сделать N итераций для выявления равенства/неравенства элементов  
M = O(1)

Класс **Polinom3**  
Содержит в себе ссылку на список многочленов (PolinomList).  
Методы:  
  
**-Polinom3(String filename)**  
Формирует PolinomList на основе файла, с именем filename. Внутри файла данные о многочлене хранятся в следующем виде:  
  
коэффициент степень1 степень2 степень3  
  
Метод считывает очередную строку, делит ее по пробелам, парсит числа и добавляет соответствующий PolinomListItem в список. Использует массив для хранения промежуточных данных.

Сложность по времени: T = O(N^2), т.к считывает все данные из файла последовательно (N раз) и для в каждой итерации добавляем данные в результирующий список при помощи add из PolinomList;

M = O(N) так как при каждом считывании создаем новый PolinomListItem и записываем его в результиющий список.

**-insert(PolinomListItem),**

**-insert(int coef, int deg1, int deg2, int deg3)**  
Работает аналогично add в PolinomList. По аналогии,  
T = O(N)  
M = O(1)  
  
**-add(Polinom3 polinom)**  
Складывает 2 полинома. Использует метод двух указателей для слияния двух упорядоченных списков многочленов  
Благодаря методу двух указателей имеет временную сложность T = O(K + N), где K и N - размер списков складываемых полиномов. Каждый элемент из списков записываются в результирующий массив в зависимости от степеней при многочлене.

Для работы использует список длины N + K, следовательно пространственная сложность  
M = O(N + K)

**-delete(int xDeg, int yDeg, int zDeg)**  
Находит и удаляет в списке элемент с соответствующими степенями. Работает аналогично delete из PolinomList.  
T = O(N); M = O(1)

**-derivative(int i)**  
Взять производную по i-ой переменной исходного полинома  
Для каждого многочлена из списка высчитываем производную по заданной переменной и записываем в новый список многочленов.  
Алгоритм вычисления производной:  
1.При помощи тернарных операторов находим нужную нам степень переменной.  
2.Если степень не равна нулю, умножаем коэффициент на текущую степень, понижаем степень и записываем получившийся многочлен в новый список. Если степень соответствующей переменной равна нулю, то считаем многочлен константой, производная которой равна нулю.  
  
Последовательно высчитывает производные и использует метод add из PolinomList. Создает новый лист, максимальная длина которого не превышает N.  
T = O(N)  
M = O(N)

**-value(int x, int y, int z)**  
Вычисляет значение полинома в точке (x,y,z).  
Последовательно вычисляет значения многочленов в точке и складывает их. Возводим в соответствующие степени переменные и перемножаем их с коэффициентом. Использует Math.pow;  
  
3 \* x^2 \* y^1 \* z^1 -> (1, 2, 3) -> 3 \* 1^2 \* 2^1 \* 3^1 = 18  
  
T = O(N)  
M = O(1)

**Метод toString**  
Чтобы понять как работает toString Polinom3, рассмотрим toString вспомогательных классов.

**PolinomItem**  
Печатает название переменной и ее степень через символ “^”(“x^2”, “y^5”,“z^1” и т.д).  
  
**PolinomListItem**  
Печатает многочлен, используя toString из PolinomItem, а так же модуль его коэффициента; Печатает знак “\*”, если PolinomItem не является первым; Если степень PolinomItem = 0, он не печатается.  
(“5 \* x^2”, “1 \* x^2 \* y^1”, “4” и т.д)

**PolinomList**  
Используется в Polinom3. Печатает весь полином. Последовательно распечатывает многочлены, используя toString из PolinomListItem. Печатает “ - “ или “ + ” между многочленами в зависимости от знака коэффициента после них.  
T = O(N)  
M = O(1)

**Тесты**

Для написания тестов была использована библиотека JUnit

|  |  |  |  |
| --- | --- | --- | --- |
| **Тестируемый класс** | **Название теста** | **Тестируемый метод** | **Краткое описание** |
| **PolinomList** | headCaseOfAdd | add | Добавление элемента в начало списка |
| **PolinomList** | addNull | add | Добавление в список null |
| **PolinomList** | replaceCaseOfAdd | add | Добавление уже существующего элемента (замена) |
| **PolinomList** | deleteHead | delete | Удаление головы списка |
| **PolinomList** | deleteFrom  Empty | delete | Удаление элемента из пустого списка |
| **PolinomList** | deleteNull | delete | Удаление null из списка |
| **PolinomList** | deleteCommon | delete | Удаление обычного элемента |
| **PolinomList** | deleteUnknown | delete | Удаление элемента, не содержащегося в списке |
| **PolinomList** | deleteTail | delete | Удаление последнего элемнета списка |
| **Polinom3** | correctFileInput | конструктор | Считывание с существующего непустого файла |
| **Polinom3** | fileNotExist | конструктор | Считываение с несуществующего файла |
| **Polinom3** | emptyFileInput | конструктор | Считывание с существующего пустого файла |
| **Polinom3** | correctAdd | add | Сложение обыкновенных полиномов |
| **Polinom3** | mergeAdd | add | Сложение полиномов, некоторые части полиномов должны быть объединены |
| **Polinom3** | emptyAdd | add | Складывание с пустым полиномом |
| **Polinom3** | common  DerivativeTest | derivative | Вычисление обычной производной |
| **Polinom3** | null  DerivativeTest | derivative | Вычисление производной, результат которой - ноль |
| **Polinom3** | empty  DerivativeTest | derivative | Вычисление производной у пустого полинома |
| **Polinom3** | valueOfNull | value | Вычисление значения в точке у пустого полинома |
| **Polinom3** | constValue | value | Вычисление значения в точке у полинома вида  C \* x^0 \* y^0 \* z^0 |
| **Polinom3** | count  CommonValue | value | Вычисление значения в точке у обычного полинома |