Контрольное мероприятие №2

Для выполнения КМ необходимо выполнить задания: 1а, 1б на удовлетворительно, дополнительно 1в, 1г, 1д, 2 — на хорошо, дополнительно 3 — на отлично.

Эмпирический анализ временной сложности алгоритмов

Для каждого n от 1 до 2000 произведите для пяти запусков замер среднего машинного времени исполнения программ, реализующих нижеуказанные алгоритмы и функции. Изобразите на графике полученные данные, отражающие зависимость среднего времени исполнения от n. Проведите теоретический анализ временной сложности рассматриваемых алгоритмов и сравните эмпирическую и теоретическую временные сложности.

1) Сгенерируйте n-мерный случайный вектор v = [v_1, v_2, ..., v_n] с неотрицательными элементами. Для полученного вектора v осуществите реализацию алгоритмов:

а) Любой алгоритм, имеющий константную сложность.

б) Алгоритм, выполняющий сумму элементов.

в) Алгоритм, выполняющий произведение элементов.

г) Алгоритм, вычисляющий значение многочлена по классической схеме:

$$P(x) = \sum_{k=1}^{n} v_k x^{k-1}$$

д) Алгоритм, вычисляющий значение многочлена по схеме Горнера:

$$P(x) = v_1 + x(v_2 + x(v_3 + \cdots))$$

2) Сгенерируйте случайные матрицы A и B размером n × n с неотрицательными элементами. Найдите обычное матричное произведение матриц A и B.

3) Проведите теоретический анализ временной анализ сложности алгоритма и сравните эмпирическую и теоретическую временные сложности алгоритма из папки «Коды к КМ-2». Номер варианта выбрать случайным образом. При проведении теоретического анализа не примитивные операции оценить отдельно.

Сложность примитивных операций

Операция	Пример	Класс сложности	Примечание
Обращение по индексу	l[i]	O(1)	
Определение длины массива	len(l)	O(1)	
Append	l.append(5)	O(1)	Почти всегда
Pop последнего элемента	l.pop() l.pop(-1)	O(1)	
Pop произвольного элемента	l.pop(i)	O(N)	O(N-i): l.pop(0):O(N)
Очистка списка	l.clear()	O(1)	Аналогично l = []
Срез	l[a:b]	O(b-a)	l[1:5]:O(l)/l[:]:O(len(l)-0)=O(N)
Расширение	l.extend(...)	O(len(...))	Зависит от длины расширения
Создание списка	list(...)	O(len(...))	Зависит от длины исходного материала
Сравнения ==, !=	l1 == l2	O(N)	
Вставка	l[a:b] = ...	O(N)	
Удаление элемента	del l[i] l.remove(...)	O(N)	В худшем случае
Проверки на вхождение	x in/not in l	O(N)	Линейный поиск в списке
Копирование	l.copy()	O(N)	Аналогично l[:]
Экстремумы	min(l)/max(l)	O(N)	Линейный поиск в списке
Реверс	l.reverse()	O(N)	
Цикл	for v in l:	O(N)	В худшем случае при отсутствии return/break в цикле
Сортировка	l.sort()	O(NLogN)	