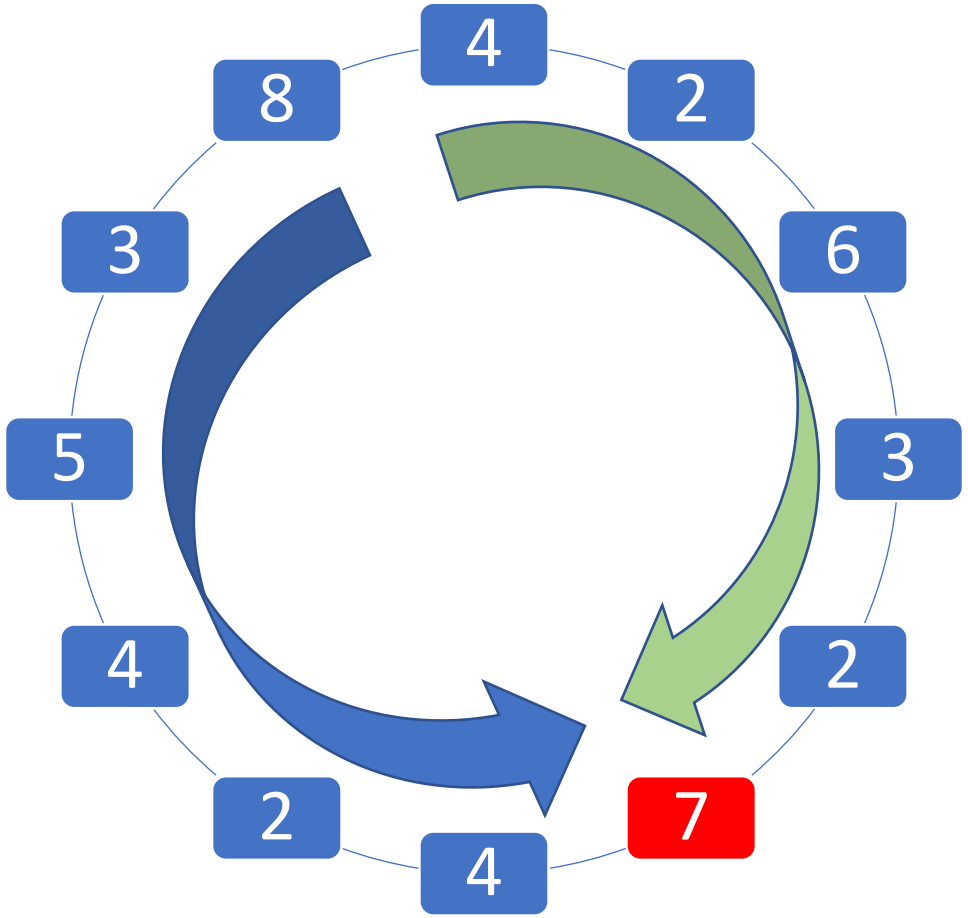
Переборные алгоритмы

на примере задачи о сборе мусора (ЕГЭ)

Формулировка задачи

Задача № 45261 с сайта «РешуЕГЭ»



Оценка

- «количество мусора в каждом пункте не превышает 1000»
- Для варианта А количество точек составляет 130 – решение можно найти «в лоб» любым алгоритмом, переполнения нет. Количество операций умножения составляет 16770, количество сравнений, сложений и прочего – примерно столько же.
- Для варианта В имеем 1200030 пунктов, что даёт общее количество умножений:

1440070800870

- Это много, плюс почти столько же операций сложений, а также сравнений (либо вычисление модуля).
- Результат выходит за диапазон целых чисел, умещается только в int64:

49113954961677

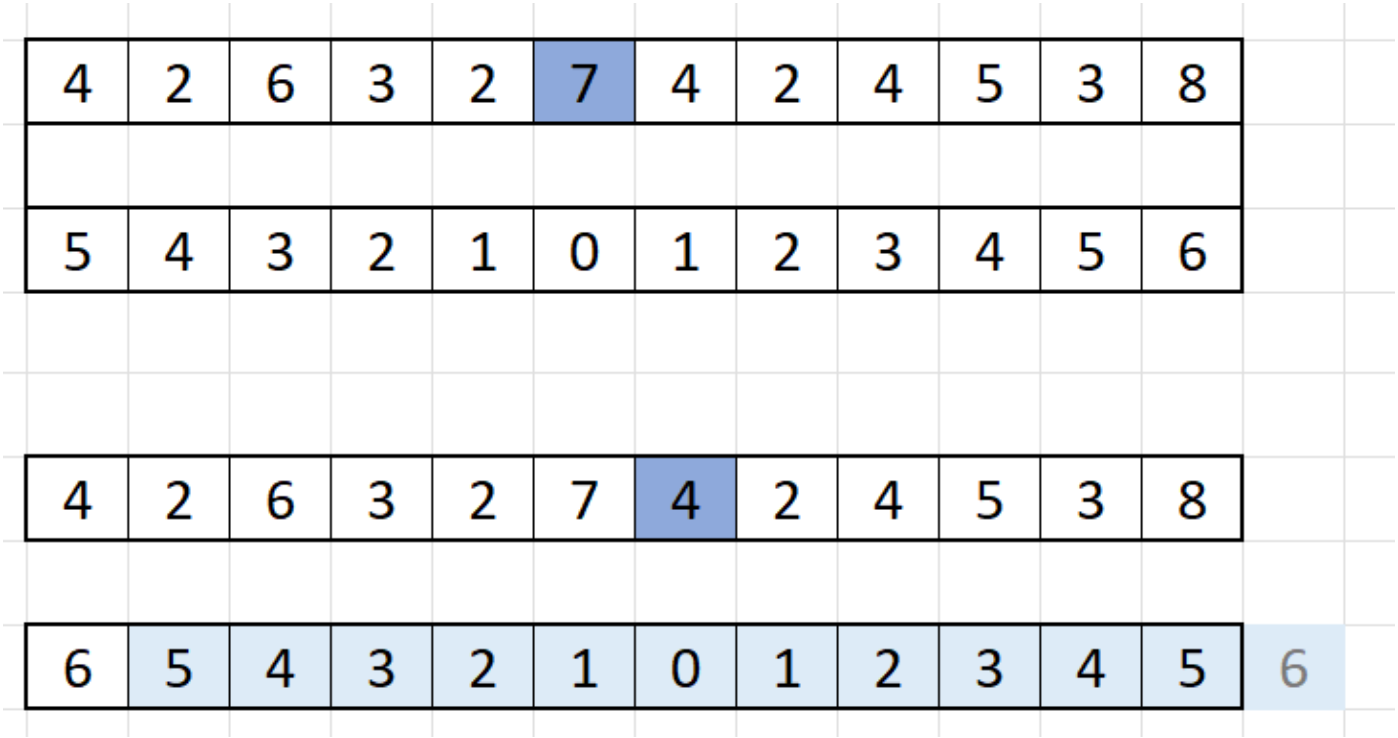
Прогноз времени

$$\frac{Time}{TotalTime} = \frac{Count}{N}$$

$$TotalTime = N\frac{Time}{Count}$$

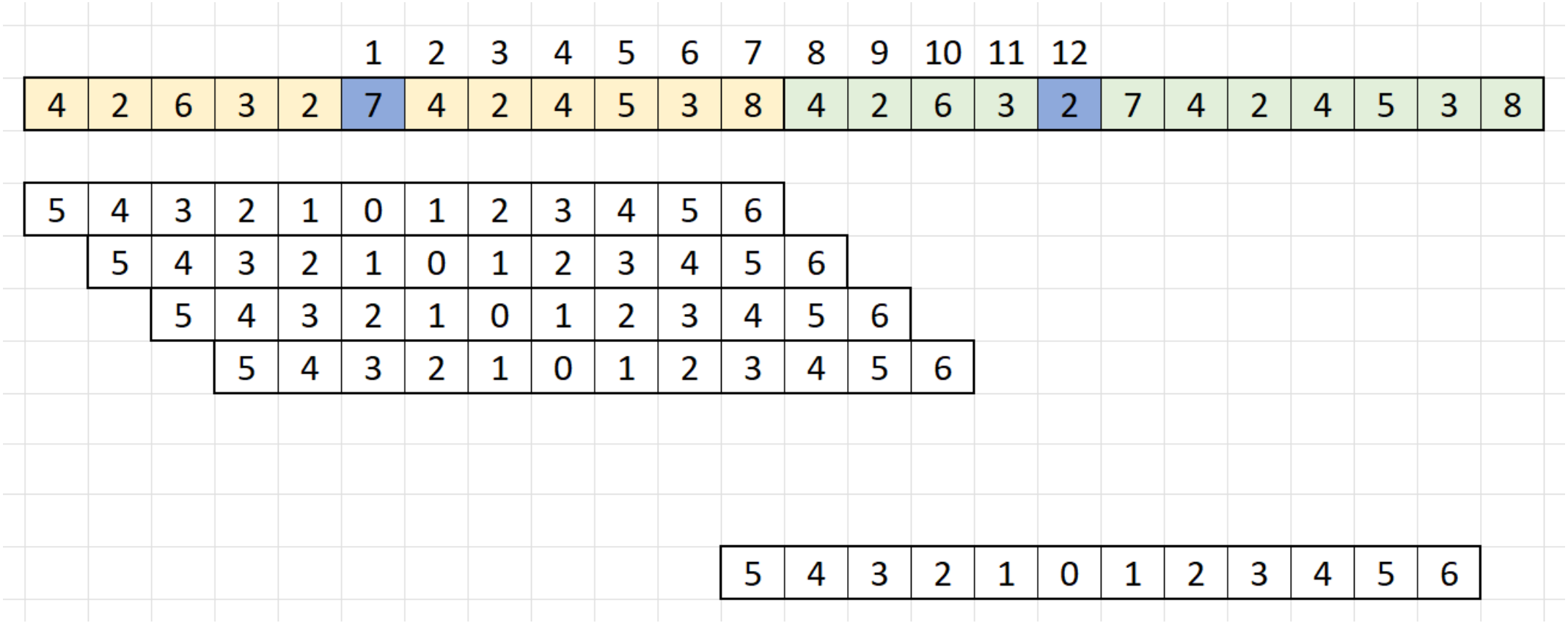
Оптимизация

По сути, имеем задачу нахождения скалярного произведения – вектор мусора на вектор расстояний. Однако остаётся проблема быстрого циклического сдвига.



Оптимизация

При удвоении вектора мусорных баков отпадает необходимость в пересчёте вектора расстояний и различных проверках.



На чем тестируем

- Ноутбук с процессором Ryzen 5 5500U – 6 ядер, 12 потоков. На полную долговременную нагрузку такие системы не рассчитаны, частота при 100% нагрузке падает до ~3.5 GHz;
- Теоретически, процессор относительно «свежий», и обеспечивает поддержку различных технологий SSE и AVX;
- Windows 11, 8 Gb RAM;
- В таких задачах объем оперативной памяти, как и её скорость, влияют на результат незначительно – все данные помещаются в кэш процессора (хотя стратегии работы на скорость всё же влияют).

Время работы в минутах

Язык	Алгоритм	Время (минуты)
Python	Наивный	7000 (> 4 суток)
Python	Оптимизация	1900 (> суток)
Python	Оптимизация + NumPy	18
Python	Оптимизация + NumPy + ThreadPool	5.9
PascalABC.NET	Наивный	60
PascalABC.NET	Оптимизация	24
PascalABC.NET	Оптимизация + OpenMP	3.5
C++	Наивный	25
C++	Оптимизация	15
C++	Оптимизация + OpenMP	3.5
C++	Оптимизация + OpenMP max	2.2

Выводы

1. Python способен решать такие задачи только с использованием NumPy (C/C++), что сложно назвать «возможностями языка». Аналоги NumPy есть для большинства языков – MKL, AOCL, BLAS (LinPack). Существенная проблема – необходимо следить за типами данных, к чему Python не приучает.
2. PascalABC.NET показал на удивление хорошие результаты с минимумом изменений в программном коде. Хотя думать всё же надо. Решается задача стандартными средствами языка, без переработки логики алгоритма.
3. C++ оказался в этой задаче на удивление «капризным», хотя лидерство всё же отстоял. Потребовал неочевидных подборов ключей оптимизации. Более того, разные результаты на разных компиляторах, ошибки работы с типами данных.
4. Сама задача оказалась вполне решаемой средствами всех трёх языков, а при вычислениях на GPU понадобится меньше минуты на решение полным перебором.
5. Все выводы могут быть поставлены под сомнение, т.к. основаны на не идеальном знании автором возможностей языков.