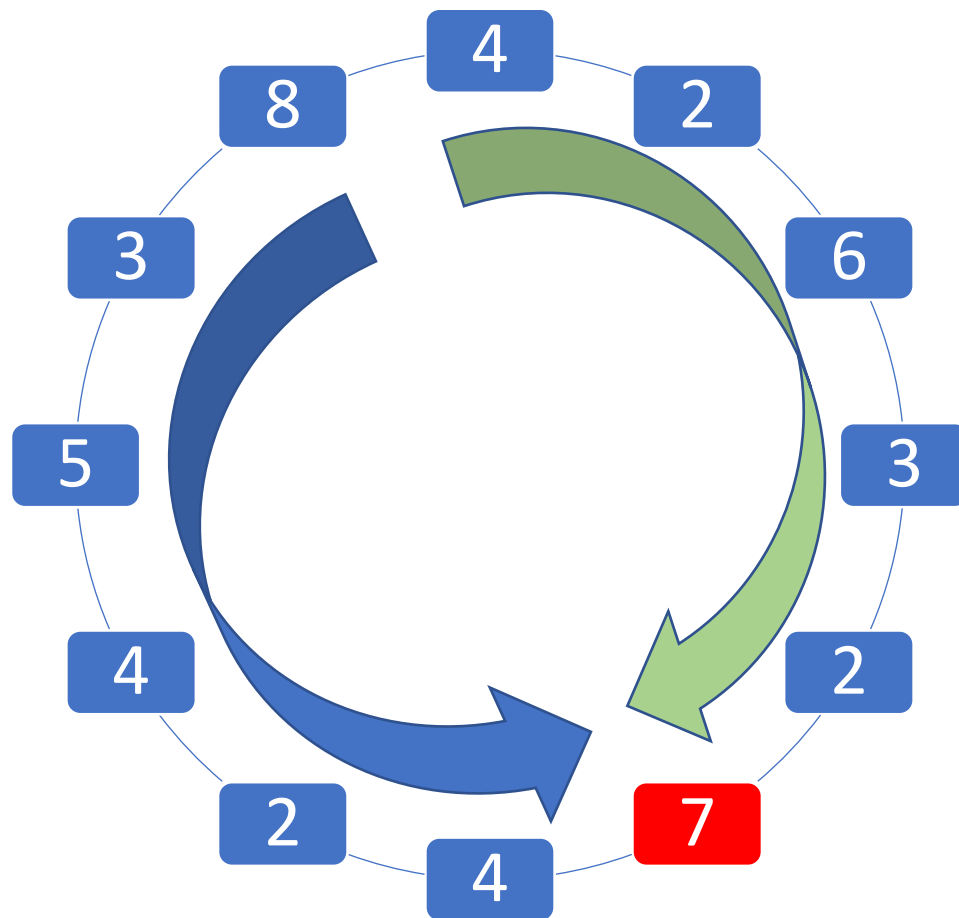


Переборные алгоритмы

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, уместается только в `int64`:

49113954961677

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, уместается 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.

4	2	6	3	2	7	4	2	4	5	3	8	
5	4	3	2	1	0	1	2	3	4	5	6	
4	2	6	3	2	7	4	2	4	5	3	8	
6	5	4	3	2	1	0	1	2	3	4	5	6

Оптимизация

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

[illegible]

Время работы в минутах

Язык	Алгоритм	Время (минуты)
Python	Наивный	7000 (> 4 суток)
Python	Оптимизация	1900 (> суток)
Python	Оптимизация + NumPy	18
PascalABC.NET	Наивный	60
PascalABC.NET	Оптимизация	24
PascalABC.NET	Оптимизация + OpenMP	3.5
C++	Наивный	60
C++	Оптимизация	49
C++	Оптимизация + OpenMP	8.5
C++	Оптимизация + OpenMP max	2.2

Выводы

1. Python способен решать такие задачи только с использованием NumPy (C/C++), которая в базовую поставку не входит. 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.
3. C++ оказался в этой задаче на удивление «капризным», хотя лидерство всё же отстоял. Потребовал неочевидных подборов ключей оптимизации.
4. Сама задача оказалась вполне решаемой, а при вычислениях на GPU понадобится меньше минуты на решение полным перебором.