

Лабораторная работа №6 по курсу дискретного анализа: динамическое программирование

Выполнил студент группы 08-303 МАИ Арусланов Кирилл

Условие

Имеется натуральное число n . За один ход с ним можно произвести следующие действия:

- Вычесть единицу;
- Разделить на два;
- Разделить на три.

Стоимость каждой операции равна текущему значению числа n . Стоимость преобразования — суммарная стоимость всех операций. Необходимо преобразовать число n в единицу так, чтобы суммарная стоимость была минимальной. Делить можно только нацело.

Формат ввода: одно число $2 \leq n \leq 10^7$

Формат вывода: минимальная суммарная стоимость и последовательность операций (-1 , $/2$, $/3$), разделённых пробелами.

Метод решения

Используется метод **динамического программирования** (вариант снизу вверх — *bottom-up*). Пусть $dp[x]$ — минимальная стоимость преобразования числа x в 1. Тогда выполняется рекуррентное соотношение:

```
dp[1] = 0
dp[x] = x + min(
    dp[x-1],
    dp[x/2] (если  $x \% 2 == 0$ ),
    dp[x/3] (если  $x \% 3 == 0$ )
)
```

При вычислении для каждого `x` запоминается выбранная операция, позволившая достичь минимальной стоимости. Восстановление пути выполняется обратным проходом от `n` к 1.

Сложность алгоритма: время — $O(n)$, память — $O(n)$.

Описание программы

Основные части:

- Массивы `dp` (тип `long long`) и `op` (тип `unsigned char`) для хранения минимальных стоимостей и кодов операций;
- Основной цикл вычисляет значения `dp[x]` для всех `x` от 2 до `n`;
- Восстановление последовательности операций идёт по массиву `op`, начиная с `n`;
- Для ускорения вывода результат аккумулируется в `std::string` и выводится одной операцией.

Дневник отладки

- Реализована рекурсивная версия с мемоизацией, которая могла бы вызвать переполнение стека;
- Переписана в итеративную форму (*bottom-up*), что позволило достичь стабильной линейной производительности;
- Оптимизирован вывод — результат формируется в строку и выводится одним вызовом.

Тест производительности

Для проверки производительности программа запускалась с различными значениями `n`. Измерялось время работы основного цикла вычисления `dp` с помощью `std::chrono::high_resolution_clock`. Результаты приведены ниже:

--

n	Время (мс)	dp[n]
100 000	0	203 703
200 000	0	403 703
500 000	2	1 008 788
1 000 000	3	2 008 788
2 000 000	7	4 008 788
5 000 000	18	10 079 033
10 000 000	23	20 079 033

Рост времени вычислений приблизительно пропорционален росту n , что подтверждает теоретическую сложность $O(n)$. Для малых n измерения неточны из-за ограниченного разрешения таймера, но при $n \geq 10^6$ наблюдается чёткая линейная зависимость.

Выводы

Разработан эффективный алгоритм на основе динамического программирования, обеспечивающий нахождение минимальной стоимости преобразования числа n в 1 за линейное время. Алгоритм показал стабильную работу на больших входных данных и подтвердил заявленную сложность. Возможные применения — оптимизация последовательностей операций, оценка стоимости вычислительных процессов и задачи минимизации ресурсов.