Лабораторна робота 4 (Складність алгоритмів)

Головата Карина МІ-41

У цій роботі було проведено кілька раундів тестування чотирьох алгоритмів обчислення CRC-16-T10-DIF: простого послідовного, табличного, дзеркального послідовного та дзеркального табличного. Кожен з алгоритмів був протестований на випадковому повідомленні довжиною 1000 біт, і було виміряно їх ефективність за часом виконання та використанням пам'яті.

Опис алгоритмів

1. **Простий послідовний алгоритм**: Виконує побітові операції з кожним символом вхідного повідомлення з використанням полінома для обчислення контрольної суми.
2. **Табличний алгоритм**: Використовує попередньо обчислені значення в таблиці замість побітових операцій, що дозволяє скоротити час обчислення за рахунок трохи більшого споживання пам'яті.
3. **Дзеркальний послідовний алгоритм**: Аналогічний до простого алгоритму, але з інверсією порядку бітів у повідомленні та результаті.
4. **Дзеркальний табличний алгоритм**: Поєднує дзеркальну обробку бітів з таблицею для пришвидшення обчислень.

Результати тестування

Алгоритм	**Середній час виконання (сек)**	**Середня використана пам'ять (MiB)**
Простий послідовний	0.45796	23.6949
Табличний	0.45656	23.6960
Дзеркальний послідовний	0.46081	23.6979

Алгоритм	Середній час виконання (сек)	Середня використана пам'ять (MiB)
Дзеркальний табличний	0.45682	23.6983

Теоретична ресурсна складність:

Аналізуємо обчислювальну складність кожного з алгоритмів. Основний акцент робимо на кількість операцій (зокрема, зсувів та XOR) та використання додаткової пам’яті.

Простий послідовний алгоритм:

- **Часова складність:** O(N * M), де N — кількість бітів у повідомленні, а M — ступінь полінома (у нашому випадку 16).
  - Для кожного біта повідомлення виконується до M операцій (зсуви, перевірка старшого біта та можливий XOR).
  - Основна частина роботи — це послідовне оброблення кожного біта повідомлення.
- **Складність за пам’яттю:** O(1)
  - Не потребує додаткової пам'яті, крім збереження поточного значення `crc` та полінома.

Табличний алгоритм:

- **Часова складність:** O(N), де N — кількість байтів у повідомленні.
  - Завдяки попередньо обчисленій таблиці на 256 значень, алгоритм обробляє кожен байт за постійну кількість операцій (1 звернення до таблиці та один XOR).
- **Складність за пам’яттю:** O(256)
  - Потребує додаткової пам'яті для збереження таблиці на 256 значень (таблиця займає 512 байт для 16-бітного значення CRC).
- Таблиця дозволяє зменшити кількість обчислень, але вимагає збереження цієї таблиці в пам’яті.

Дзеркальний послідовний алгоритм:

- **Часова складність:** O(N * M), аналогічно простому послідовному алгоритму.
  - Додається операція інверсії бітів, що має постійну складність O(M) для 16-бітного числа.
- **Складність за пам’яттю:** O(1)
  - Використовується додаткова пам'ять лише для інверсії бітів.

Дзеркальний табличний алгоритм:

- **Часова складність:** O(N)
  - Як і у табличному алгоритмі, завдяки використанню попередньо обчисленої таблиці обробляється кожен байт за постійну кількість операцій.
  - Інверсія порядку бітів виконується як додаткова операція з постійною складністю.
- **Складність за пам’яттю:** O(256)
  - Потребує збереження таблиці та інверсії результату.

Час виконання:

- **Найшвидшим** виявився **табличний алгоритм**, який виконався в середньому за **0.45656 секунд**. Це узгоджується з теорією, оскільки використання попередньо обчислених значень у таблиці значно пришвидшує обчислення, скорочуючи кількість побітових операцій. Таблиця дозволяє уникнути обробки кожного біта окремо, що суттєво зменшує кількість обчислень.
- **Найповільнішим** виявився **дзеркальний послідовний алгоритм**, із середнім часом виконання **0.46081 секунд**. Це також очікувано, оскільки інверсія бітів додає додаткові операції до обробки кожного біта, що дещо уповільнює алгоритм порівняно з іншими.
- **Дзеркальний табличний алгоритм** та **простий послідовний алгоритм** показали проміжні результати — **0.45682 секунд** та **0.45796 секунд** відповідно. Це свідчить про те, що хоча дзеркальний табличний алгоритм має додаткові витрати на інверсію бітів, його швидкість обчислення залишається високою завдяки використанню таблиці. Простий послідовний алгоритм, натомість, трохи

поступається в швидкості через необхідність обробляти кожен біт без використання таблиці.

Використана пам'ять:

- Усі алгоритми показали **дуже подібне споживання пам'яті**, приблизно **23.6949-23.6983 MiB**. Це свідчить про те, що жоден із методів не потребує значних додаткових ресурсів пам'яті, оскільки основне споживання пам'яті припадає на сам процес виконання обчислень і зберігання проміжних значень.
- **Простий послідовний та табличний алгоритми** споживали трохи менше пам'яті порівняно з дзеркальними аналогами. Це пов'язано з відсутністю операцій інверсії бітів, які вимагають додаткової пам'яті для збереження інвертованих значень.
- **Дзеркальний табличний алгоритм** та **дзеркальний послідовний алгоритм** використовують трохи більше пам'яті — **23.6983 MiB** та **23.6989 MiB** відповідно. Це пов'язано з додатковими витратами на інверсію бітів у процесі обчислення, що потребує зберігання проміжних результатів.

Висновки:

1. **Табличний алгоритм** показав найкращі результати за часом виконання, що робить його оптимальним вибором для випадків, коли потрібно обробляти великі обсяги даних.
2. **Дзеркальний послідовний алгоритм** виявився найповільнішим, оскільки додаткові операції інверсії бітів впливають на загальний час обчислення. Його доцільно використовувати у випадках, коли потрібне оброблення даних у дзеркальному форматі, наприклад, для сумісності з певними стандартами передачі даних.
3. **Дзеркальний табличний алгоритм** показав результат, близький до табличного, що свідчить про його ефективність навіть при наявності інверсії бітів. Він поєднує швидкість табличного методу з можливістю обробляти дзеркально відображені дані.
4. Усі алгоритми мають **подібне споживання пам'яті**, що робить їх використання можливим у різних середовищах без значного впливу на ресурси. Невелика різниця в споживанні пам'яті між дзеркальними та недзеркальними алгоритмами свідчить про незначний вплив операцій інверсії бітів на загальну пам'ять.