Программирование

Сергей Салищев

Тема 3. Способы доступа к данным

Обзор

- Зачем нужны эффективные структуры данных
- Отличие способов доступа и организации данных
- Способы физической организации данных
- Иерархическая память
- Одновременный доступ
- Основные способы доступа к данным

Зачем нужны эффективные структуры данных

- Программа работает с данными
  - Входные данные
  - Выходные данные
  - Промежуточные данные
- Операции доступа к данным — базовые блоки более сложных программ
  - Могут оказывать доминирующее влияние на скорость работы

Отличие способов доступа и организации данных

- Различные способы доступа к данным
  - Связь между данными
  - Связь между операциями
- Математическая или житейская метафора
  - Пример: вектор, множество, очередь
- Ограничения
  - Пример: разреженность, отношение чтения/записи
- Один и тот же способ доступа, разная организация данных
  - Различная эффективность
    - На разных операциях
    - При разных ограничениях

Способы физической организации данных

- Память с прямой адресацией
  - Косвенность
  - Блочный доступ
- Внешние хранилища данных
  - Диски, ленты
- Потоки
- Экономия памяти → эффективность

Иерархия памяти

- Провода
- Регистры
- Память
  - Расстояние ⇔ размер, скорость
- Кэши
  - Когерентность
- Внешние хранилища
  - Виртуализация памяти и выгрузка

Одновременный доступ

- Упорядочивание операций
  - блокировки
- Гранулярность блокировок
- Самосинхронизированные структуры данных

Основные способы доступа к данным

- Магазин (Stack)
- Очередь (Queue)
- Итератор (Iterator)
- Список (List)
- Очередь с приоритетами (PriorityQueue)
- Граф (Graph)
- Строка (String)
- Отображение (Map)
- Вектор (Vector)
- Матрица (Matrix)
- Множество (Set)
- Упорядоченное множество (Sorted set)
- Поток (Stream)
- Кэш ресурсов (LRU)
- Десятичная дробь (Decimal)

Отображение (Map)

- Операции
  - взять(ключ), установить(ключ, значение), удалить(ключ)
- Лучшая асимпт. сложность - O(1)*
- Реализации
  - Хэш-таблица, Дерево, другие*
- Библиотечные классы
  - Map*, HashMap
- Любой способ доступа может быть представлен как отображение*

Вектор (Vector)

- Операции
  - взять(i), присвоить(i, значение)
- Лучшая асимпт. сложность - O(1)
- Реализации
  - Массив, динамический массив, файл
- Библиотечные классы
  - ArrayList, RandomAccessFile

Матрица (Matrix)

- Операции
  - взять(i0, … , iN),
  - присвоить(i0, … , iN, значение)
- Лучшая асимпт. сложность - O(1)
- Реализации
  - Вектор*, Вектор векторов*
- Библиотечные классы
  - ArrayList

Итератор (Iterator)

- Операции
  - следующий
- Лучшая асимпт. сложность - O(1)
- Реализации
  - Все
- Библиотечные классы
  - Iterator*, Iterable*

Магазин (Stack)

- Операции
  - положить(значение), взять
- Лучшая асимпт. сложность - O(1)
- Реализации
  - Список, массив, динамический массив
- Библиотечные классы
  - ArrayList, LinkedList

Очередь (Queue)

- Операции
  - поставить(значение), взять
- Лучшая асимпт. сложность - O(1)
- Реализации
  - Список, циклический буфер, канал ввода/вывода
- Библиотечные классы
  - List*, ArrayList, LinkedList

Список (List)

- Операции
  - следующий, предыдущий, вставить(значение), удалить
- Лучшая асимпт. сложность - O(1)
- Реализации
  - Односвязный список, двусвязный список
- Библиотечные классы
  - LinkedList, ListIterator*

Множество (Set)

- Операции
  - проверить(значение), добавить(значение), удалить(значение)
- Лучшая асимпт. сложность - O(1)*
- Реализации
  - Хэш-таблица, Дерево
- Библиотечные классы
  - Set*, HashSet

Упорядоченное множество (SortedSet)

- Операции
  - проверить(значение), добавить(значение), удалить(значение), большие(значение), меньшие(значение)
- Лучшая асимпт. сложность — O(log N)*
- Реализации
  - Дерево, сортированный массив
- Библиотечные классы
  - SortedSet*, SortedMap*, ArrayList

Строка (String)

- Операции
  - +, >, =, подстрока(начало, конец), найти(подстрока), заменить(подстрока)
- Лучшая асимпт. сложность — O(N)*
- Реализации
  - Массив, динамический массив, буфер с дыркой, дерево
- Библиотечные классы
  - String, StringBuilder

Десятичная дробь (Decimal)

- Операции
  - +, -, *, /, >, =
- Лучшая асимпт. сложность — O(log N), $O(\log^2 N)$ для */
- Реализации
  - Целое число, вектор*
- Библиотечные классы
  - BigDecimal

Кэш ресурсов (LRU)

- Операции
  - взять(ключ), добавить(ключ, значение),
- Лучшая асимпт. сложность - O(1)*
- Реализации
  - Хэш-таблица + список,
- Библиотечные классы
  - LinkedHashMap

Очередь с приоритетами (Priority Queue)

- Операции
  - добавить(задание, приоритет), взять
- Лучшая асимпт. сложность — O(log N)
- Реализации
  - Куча, массив списков
- Библиотечные классы
  - PriorityQueue

Поток (Stream)

- Операции
  - прочитать/записать
- Лучшая асимпт. сложность - O(1)*
- Реализации
  - Массив, файл, канал
- Библиотечные классы
  - InputStream*, OutputStream*, Reader*, Writer*

Граф (Graph)

- Операции
  - добавить(вершина/дуга), удалить(вершина/дуга), проверить(дуга), соседи(вершина), покрасить(вершина/дуга)
- Лучшая асимпт. сложность - O(1)*
- Реализации
  - Матрица, список списков
- Библиотечные классы
  - List*, HashMap

Вопросы?