СТРУКТУРА ДАННЫХ РАЗРЕЖЕННАЯ ТАБЛИЦА

И. З. Искандаров
Ургенчский филиал Ташкенсткого университета информационных технологий имени Мухаммада ал-Хорезмий
islom.iskandarov@ubtuit.uz

АННОТАЦИЯ

В данной статье рассматривается структура данных разреженная таблица: описание, возможности, примеры применения к задачам.

Ключевые слова: структуры данных, алгоритмы, разреженная таблица

ABSTRACT

This article discusses the data structure sparse table: description, features, examples of application to tasks.

Keywords: data structures, algorithms, sparse table

ВВЕДЕНИЕ

Структура данных (data structure) — это способ хранения и организации данных, облегчающий доступ к этим данным и их модификацию[1]. Каждая структура данных имеет свою сферу применения, то есть типы задач, которые может решать. В этой статье рассмотрим структуру данных разреженная таблица (англ. Sparse table), которая хранит данные в особой(разреженной) форме и позволяет быстро отвечать на запросы в отрезке без модификаций элементов.

АНАЛИЗ ЛИТЕРАТУРЫ И МЕТОДОЛОГИЯ

Данная структура данных была представлена в статье The LCA Problem Revisited[2] как часть оптимизированного решения задачи LCA (*Least Common Ancestor*). Задача LCA состоит в следующем: Пусть дано дерево G. На вход поступают запросы вида (V1, V2), для каждого запроса требуется найти их наименьшего общего предка, т.е. вершину V, которая лежит на пути от корня до V1, на пути от корня до V2, и из всех таких вершин следует выбирать самую нижнюю. Решение задачи представленная в этой статье связана с задачей RMQ (*Range Minimum Query*) нахождения минимума на отрезке и структура данных разреженная таблица позволяет эффективно решать эту задачу. В книге Competitive

Programmer's Handbook[3] в разделе *Запросы статичных массивов* также показан решение для запроса минимумов с примером с статичным массивом. В интернет ресурсах, посвящённых структурам данных и алгоритмам тоже представлен данный алгоритм. Например, в [4] представлен данный алгоритм с реализацией в языке C++, более подробно рассмотрена тема в [5] в том числе применение для коммутативных операций.

Постановка задачи. Предположим, что у нас есть массив чисел A состоящий из N элементов и нам нужно найти определённое значение для некоторого отрезка массива A. Значениями могут быть, например: минимальный элемент на отрезке, максимальный элемент на отрезке, сумма и т.д. Разреженная таблица позволяет с предпосчётом за время и с затратами памяти $O(NlogN)$ для идемпотентных операций(минимум, максимум, НОД и т.д.) отвечать на запрос за время $O(1)$. Далее рассмотрим, как пример использование разреженной таблицы для нахождения минимального элемента на отрезке в запросах.

Описание структуры. Идея структуры данных разреженная таблица — это использование таблицы T для хранения ответов для отрезков. Но мы не можем хранить значения для всех отрезков так как расход памяти составит $O(N^2)$, а будем хранить значения для отрезков длины 2^k где $0 \le k \le \lfloor \log_2 n \rfloor$. Формально в *T[l][j]* хранится значения для отрезка с левой границей l длины 2^j, то есть значение для отрезка с индексами $[l; l + 2^j - 1]$. Например для массива с элементами [3, 2, 4, 5, 1, 1, 5, 3] длины 8 таблица будет выглядеть следующим образом:

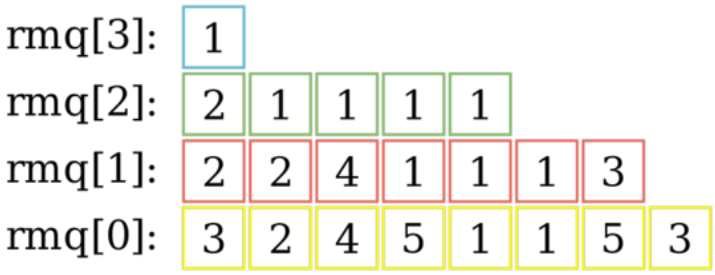

Рис 1. Структура таблицы для 8 элементов

Построение. Рассмотрим построение для хранения значения минимального элемента на отрезке. В первую очередь нам нужно вычислить значения двоичных логарифмов их мы будем рассчитывать целыми числами с округлением вниз. Сделаем предпосчёт и сохраним значения в массиве:

```
1. int lg[N + 1];
2. lg[1] = 0;
3. for (int i = 2; i <= N; ++i) {
```

```
    lg[i] = lg[i / 2] + 1;
}
```

Листинг 1. Расчёт значений двоичных логарифмов чисел

Следующим шагом сделаем построение самой таблицы:

```
// Чтение размера массива N и массива A
// ...

  for (int i = 1; i <= N; ++i) {
       T[i][0] = A[i];
  }
  for (int i = 1; (1 << i) <= N; ++i) {
    int len = (1 << i);
    for (int j = 1; j + len - 1 <= N; j++) {
       int tail = j + len - 1;
       T[j][i] = min(T[j][i - 1], T[tail - len / 2 + 1][i - 1]);
    }
  }
```

Листинг 2. Построение разреженной таблицы для минимума

Получение ответов. Для получения значения для отрезка $[l;r]$ “объединим” ответы для двух отрезков $[l;l+2^j-1]$ и $[r-2^j+1;r]$, где j наибольшее целое число для которого $2^j \le r-l+1$,
то есть $j=\lfloor \log_2(r-l+1) \rfloor$.

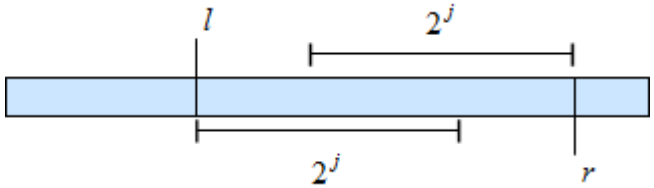

Рис 2. Объединение отрезков для ответа на запрос

```
int get(int l, int r) {
    int len = r - l + 1;
    int p = lg[len];

    return min(T[l][p], T[r - (1 << p) + 1][p]);
}
```

Листинг 3. Получение ответа на запрос минимума в отрезке

Далее в запросах можем использовать эту функцию:

```
int Q;
cin >> Q;
for (int i = 1; i <= Q; i++) {
    int l;
    int r;
    cin >> l >> r;
    cout << get(l, r) << "\n";
}
```

Листинг 4. Использование разреженной таблицы в запросах

Указанный выше пример можем использовать для других идемпотентных операций, например, для нахождения максимума, НОД и т.п. изменив функцию расчёта при построении таблицы и получении ответа.

РЕЗУЛЬТАТЫ И ОБСУЖДЕНИЕ

Приведём сравнительный анализ разреженной таблицы и других структур данных для различных задач(запросов) и размера данных. Метрикой сравнения возьмём время исполнения программ с использованием этих структур данных на языке C++. В расчёт входит суммарное потраченное время на выполнение следующих операций: построение структуры, чтение запросов с файла, вычисление и вывод результатов на файл. Размер массива обозначим через N, число запросов равна Q.

Структура / Размер данных	$N = 10^4$ $Q = 10^4$	$N = 10^5$ $Q = 10^5$	$N = 10^5$ $Q = 10^6$	$N = 10^6$ $Q = 5 * 10^6$
Дерево отрезков	56 мс	619 мс	5754 мс	34178 мс
SQRT-декомпозиция	63 мс	806 мс	7235 мс	58415 мс
Разреженная таблица	67 мс	802 мс	5594 мс	30761 мс

Таблица 1. Сравнение структур для задачи RMQ

Структура / Размер данных	$N = 10^4$ $Q = 10^4$	$N = 10^5$ $Q = 10^5$	$N = 10^5$ $Q = 10^6$	$N = 10^6$ $Q = 5 * 10^6$
Дерево отрезков	57 мс	613 мс	6387 мс	32323 мс
SQRT-декомпозиция	56 мс	762 мс	7058 мс	58890 мс
Разреженная таблица	63 мс	722 мс	5586 мс	31661 мс

Таблица 2. Сравнение структур для задачи нахождения НОД на отрезке

Структура / Размер данных	$N = 10^4$ $Q = 10^4$	$N = 10^5$ $Q = 10^5$	$N = 10^5$ $Q = 10^6$	$N = 10^6$ $Q = 5 * 10^6$
Дерево отрезков	58 мс	634 мс	6863 мс	40786 мс
SQRT-декомпозиция	56 мс	890 мс	8863 мс	86058 мс
Разреженная таблица	123 мс	722 мс	5922 мс	33616 мс

Таблица 3. Сравнение структур для задачи нахождения суммы на отрезке

Исходя из данных приведённых выше можем заметить, что разреженная таблица даёт выигрыш при большем количестве запросов относительно размера входных данных. Первые две таблицы относятся к задачам, для которых данный алгоритм вычисляет ответ за константное время. Дополнительно в третьей таблице дано ассоциативная операция (сумма). Для таких операций структура вычисляет ответ за логарифмичное время. Примечательно что в этом случае разреженная таблица выдаёт хорошие результате при больших размерах данных особенно в последнем случае даёт значительное преимущество.

ВЫВОД

Структура данных разреженная таблица целесообразно использовать в следующих случаях:

1. Отсутствует модификация элементов, то есть значения элементов массива не меняются.
2. Число запросов большое
3. Нужно найти значение идемпотентной функции для отрезка таких как минимум, максимум и т.п. Для таких операций разреженная таблица выдаёт ответа за время $O(1)$, в остальных случаях за $O(logN)$ что уже не даёт преимущества перед другими структурами данных.

REFERENCES

1. Алгоритмы: построение и анализ, 2-е издание. Кормен, Томас Х., Лейзерсон, Чарльз И., Ривест, Рональд Л., Штайн, Клиффорд. Пер. с англ. — М.: Издательский дом "Вильямс", 2005. — 1296 с.
2. http://www2.compute.dtu.dk/courses/02282/2021/nca/CPbook.pdf
3. M. A. Bender, M. Farach-Colton. The LCA problem revisited. In Latin American Symposium on Theoretical Informatics, 88–94, 2000.
4. https://ru.algorithmica.org/cs/range-queries/sparse-table/
5. https://peltorator.ru/posts/sparse_table/