Практика по алгоритмам #3

Содержание

1. Теория

1.1. В предыдущих сериях

Мы умеем строить частичные суммы, минимум на стеке, минимум на очереди. Мы умеем сортировать массив, у нас есть бинарный поиск, хеш-таблица. У нас есть бинарная куча.

1.2. Минимум на очереди #2

Задача: нужна структура данных с операциями `get_min`, `pop_front`, `push_back` за амортизированное $\mathcal{O}(1)$. Решение: будем хранить m_1 — позиция минимума на всей очереди, m_2 — позиция минимума среди элементов правее m_1 и так далее:
$a[m_1] = min[a_L..a_R], a[m_{i+1}] = min[a_{m_i+1}..a_R]$. Заметим, что $a[m_i] \le a[m_{i+1}]$.

Операция `get_min`	`return a[m₁];`
Операция `pop_front`	`if (L == m₁) m.pop_front();`

Операция `push_back`	`while (m.size() > 0 and a[m.last()] <= x)`
	`   m.pop_back();`
	`m.push_back(x);`

2. Задачи

1. Дана последовательность чисел: $x_1 = a,\ x_{i+1} = f(x_i)$, найти с $O(1)$ дополнительной памяти длину периода T и предпериода L за время $\mathcal{O}(L+T)$.
2. Даны два выражения с операциями `+`,`-`,`*` и скобками. Суммарная длина не превосходит 10^6. Проверить, равны ли значения выражений.
   a) Только числа. Промежуточные результаты — произвольные числа.
   b) Есть переменные, произвольное количество. Проверить тождественное равенство.
3. Частичные суммы:
   a) Много раз сделать `+=` на отрезке, в конце один раз вывести массив.
   b) Сперва много раз `+=` на отрезке, затем много раз "сумма на отрезке".
   c) В каждой целой точке x числовой прямой есть $f[x]$, изначально равная нулю. Те же запросы, что и в предыдущем пункте. Координаты запросов целые от 0 до 10^{18}.
4. Много запросов вида `color(l, r, c)` "покраска отрезка `[l..r]` массива в цвет `c`". В конце нужно одим раз вывести массив. Решение в offline.
5. За линейное время найти подотрезок массива.
   a) С максимальной суммой.
   b) С максимальной суммой, длины от L до R.
   c) С максимальной суммой, содержащий не менее k различных чисел.
6. Задачи на стек.
   a) В массиве найти для каждого элемента ближайших меньших соседей слева и справа за $O(n)$.
   b) Дана матрица из нулей и единиц. Найти наибольший по площади подпрямоугольник, состоящий только из нулей за $O(n^2)$.

7. Даны два сортированных массива длины n. Без дополнительного предподсчета найти k-ю порядковую статистику в объединении массивов.
   a) За $\mathcal{O}(\log^2 n)$.
   b) За $\mathcal{O}(\log n)$.
8. Даны два массива a и b длины n, сгенерировать все попарные суммы $a_i + b_j$ в сортированном порядке.
   a) За $\mathcal{O}(n^2 \log n)$.
   b) За $\mathcal{O}(n^3)$ с использованием $\mathcal{O}(n)$ дополнительной памяти.
   c) За $\mathcal{O}(n^2 \log n)$ с использованием $\mathcal{O}(n)$ дополнительной памяти.

3. Разобранное на паре

1. **Поиск периода и предпериода можно сделать в три шага:**
   a) `x = 0, y = 1; while (p[x] != p[y]) x += 1, y += 2;`
   Точка в которой мы остановились лежит на цикле. Обозначим ее `a`.
   b) `x = a + 1; while (p[x] != p[a]) x += 1;`
   Мы остановились в точке `a + T` ⇒ `T = x - a;`
   c) `x = 0, y = T; while (p[x] != p[y]) x += 1, y += 1;`
   Мы остановились, когда `x` попал в начало цикла ⇒ `L = x;`.
2. **Задача про разбор выражений:**
   a) Посчитаем оба выражения по модулю $10^9 + 7$.
   b) Посчитаем оба выражения по модулю 10^9+7. А в переменные подставим случайные значения. Получили хороший вероятностный алгоритм. Корректность следует из леммы Шварца-Зиппеля.
   c) Решим задачу про матрицы: проверить за $\mathcal{O}(n^2)$, что $A \cdot B = C$. Решение: выбираем случайный вектор x, и проверяем, что $A(Bx) = Cx$.
3. **Задача про покраску отрезков:**
   a) Можно решать за $\mathcal{O}(n + q\log n)$ деревом отрезков, но проще по-другому...
   b) Можно идти слева направо, обрабатывать события “отрезок начался”, “отрезок закончился” и хранить `set` открытых отрезков. Асимптотика решения также $\mathcal{O}(n+q\log n)$.
   c) Можно красить, начиная с последнего запроса, тогда однажды покрашенную клетку никогда не нужно перекрашивать, поэтому есть решение с СНМ (DSU):

```
int get(int v) { return p[i] == i ? i : get(p[i]); } // DSU
for (i = 0; i <= n; i++) // elements of array
  p[i] = i, next[i] = i + 1, color[i] = -1;
for (int q - 1; i >= 0; i--) { // queries
  int left = get(l[i]);
  while (left <= r[i]) {
    if (color[left] == -1) color[left] = c[i]; // color it!
    left = p[left] = get(next[left]); // join
  }
}
```

   Асимптотика решения: $\mathcal{O}((q + n)\mathcal{A}^{-1}(q, n))$

4. **Задача про частичные суммы:**
   `a[r+1] -= 1, a[l] += 1;`, в конце `for i: a[i] += a[i+1]`.
5. **Задача про отрезок максимальной суммы:**
   a) Жадно движемся вперед двумя указателями `l` и `r`,
      поддерживаем сумму текущего отрезка.
```
result = -infinity, sum = 0, l = 0;
for (r = 0; r < n; r++) {
    sum += a[r];
    if (sum > result) result = sum, res_l = l, res_r = r;
    if (sum < 0) sum = 0, l = r + 1;
}
```
   b) Храним очередь возможных левых концов, правый конец перебираем.
```
for (i = 0; i < n; i++)
    sum[i+1] = sum[i] + a[i]; // насчитали частичные суммы
QueueWithMin q; // все операции за амортизированное O(1)
int min_l = -R, max_l = -L; // q = sum[min_l..max_l]
for (r = 0; r <= n; r++) {
    if (!q.empty() && sum[r] - q.min() > result)
        result = sum[r] - q.min(), res_l = q.min_pos(), res_r = r;
    if (min_l++ >= 0) q.pop_front();
    if (++max_l >= 0) q.push_front(pair(sum[max_l], max_l));
}
```
   c) Для каждого `r` поддерживаем максимальное `l` такое, что отрезок `[l..r]` содержит хотя бы `k` различных чисел. Для отрезка `[l..r]` поддерживаем хеш-таблицу `count[x]` — сколько раз встречалось число `x` на отрезке `[l..r]`.
6. **Задачи на стек, пункт (а):**
```
Stack c;
a[n++] = -1; // чтобы у всех был ближайший слева меньший
for (i = 0; i < n; i++) {
  while (a[c.top()] > a[i])
    next_less[c.pop()] = a[i];
  c.push(i);
}
```

4. Домашнее задание

4.1. Обязательная часть

• Задачи, в которых нужно решение и доказательство корректности.

1. (1) Найти подотрезок с максимальным средним арифметическим элементов за $\mathcal{O}(n)$.
2. (1) Найти подотрезок с максимальной суммой, длины от L до R, содержащий от A до B различных чисел за $\mathcal{O}(n)$.
3. (2) Дан массив из $2n$ чисел. Найти минимальное **И** максимальное за $3n-2$ сравнения.
4. (3) Найти максимальную по длине подстроку, являющуюся правильной скобочной последовательностью за $\mathcal{O}(n)$.

• Задачи, в которых нужно только описание решения.

5. (2) Практика.6b.
6. (1) Практика.7a.
7. (2) Практика.7b.

• Задачи, в которых нужно только написать код ная языке `C/C++`.

8. (2) Множество и мультимножество можно хранить в виде отсортированного массива. Даны два множества A и B в отсортированном виде, за $\mathcal{O}(|A|+|B|)$ построить в таком же виде их
   a) множество-разность. Пример: $\{1,2,3\} \setminus \{2,4\} = \{1,3\}$.
   b) множество-объединение. Пример: $\{1,2,3\} \cup \{2,4\} = \{1,2,3,4\}$.
   P.S. Если вы пишете в LaTeX (а уже пора ;), попробуйте заодно пакет `\usepackage{minted}`. В `dropbox` есть пример. За подробной справкой можно обращаться ко мне и Диме Лапшину.

4.2. Дополнительная часть

Во всех задачах нужно описание решения и доказательство корректности.

1. (3) Посчитать число подпрямоугольников, состоящих только из нулей за $\mathcal{O}(n^2)$.
2. (3) Даны m сортированных массивов длины n. Нужно без дополнительного предподсчета найти k-ю порядковую статистику за время $\mathcal{O}(m \log m \log n)$.
3. (3) Придумайте, как в "очереди с максимумом" победить амортизацию. То есть, придумайте аналогичную структуру данных с временем работы $\mathcal{O}(1)$ в худшем случае на каждую операцию.