Олимпиадное программирование

Занятие 11. Динамическое программирование. НВП, НОП, НОВП, расстояние Левенштейна

Труфанов Павел Николаевич

Онлайн-школа  Фоксфорд

Foxford.ru 2019-2020

Наибольшая возрастающая подпоследовательность

Подпоследовательность массива - это последовательность чисел, которая получается удалением некоторых чисел из исходного массива.
Задача - найти наибольшую возрастающую подпоследовательность.
Требуемое время - $O(n^2)$.
dp[i] - НВП на префиксе длиной i.

Наибольшая возрастающая подпоследовательность

Требуется решить задачу за $O(n \log n)$.
Здесь необычная динамика. dp[i] - минимальное число, на которое оканчивается возрастающая подпоследовательность длины i.
Мы будем обрабатывать элементы массива по очереди и обновлять массив динамики.

Итоговый код

```
for (int i = 0; i < n; ++i) {
    d[lower_bound(d.begin(), d.end(), a
      d.begin()] = a[i];
}
for (int i = n; i >= 0; --i) {
    if (d[i] != INF) {
        cout << i << endl;
        break;
    }
}
```

Наибольшая общая подпоследовательность

Требуется найти наибольшую по длине общую подпоследовательность двух массивов. Требуется решить за $O(nm)$.
Динамика - dp[i][j] - НОП для префикса длиной i первой последовательности и префикса длиной j второй последовательности.
if a[i] == b[j]: dp[i][j] = dp[i - 1][j - 1] + 1 else: dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])

Расстояние Левенштейна

Даны две строки. За один ход мы можем удалить один символ из первой строки за *deleteCost* монет, добавить один символ в первую строку за *insertCost* монет и поменять один символ за *replaceCost* монет. Требуется за минимальную стоимость получить из первой строки вторую. dp[i][j] - расстояние Левенштейна для префикса длины i первой строки и префикса длиной j второй строки.

Переходы

$$D(i,j) = \begin{cases} 0 & ;\ i = 0,\ j = 0 \\ i * deleteCost & ;\ j = 0,\ i > 0 \\ j * insertCost & ;\ i = 0,\ j > 0 \\ D(i-1, j-1) & ;\ S_1[i] = S_2[j] \\ \min (& \\ \quad D(i, j-1) + insertCost & \\ \quad D(i-1, j) + deleteCost & ;\ j > 0,\ i > 0,\ S_1[i] \neq S_2[j] \\ \quad D(i-1, j-1) + replaceCost & \\ ) & \end{cases}$$

НОВП

Требуется найти наибольшую общую возрастающую подпоследовательность.
Решения за $O(n^4), O(n^3), O(n^2)$

Четвертая степень

d[i][j] - НОВП для префиксов длины i и длины j, при этом эти элементы обязательно входят в НОВП.
Если a[i] != b[j], то d[i][j] = 0
Иначе d[i][j] = max(d[i1][j1]) + 1 (i1 < i, j1 < j)

Третья степень

d[i][j] - НОВП для префиксов длины i и длины j, при этом элемент a[i] обязательно входит в НОВП.
Если a[i] != b[j], то d[i][j] = d[i][j - 1]
Иначе d[i][j] = max(d[i1][j - 1]) + 1 (i1 < i, a[i1] < a[i])

Оптимальное решение

d[i][j] - НОВП для префиксов длины i и длины j, при этом элемент b[j] обязательно входит в НОВП
Первый случай: a[i] не входит в НОВП, тогда d[i][j] = d[i - 1][j]
Второй случай: a[i] входит в НОВП. Тогда a[i] = b[j], следовательно d[i][j] = max(d[i - 1][j1]) + 1 (j1 < j, b[j1] < b[j]=a[i]). Но заметим, что при фиксированном i мы можем поддерживать такое оптимальное j1, что d[i - 1][j1] максимально и b[j1] < b[j] = a[i])

До встречи!

FOXFORD.RU

Онлайн-школа Фоксфорд

Фоксфорд