Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №7 по курсу «Дискретный анализ»

Студент: А. А. Почечура
Преподаватель: А. А. Кухтичев
Группа: М8О-306Б
Дата:
Оценка:
Подпись:

Москва, 2023

Лабораторная работа №7

Задача: Задан прямоугольник с высотой n и шириной m, состоящий из нулей и единиц. Найдите в нём прямоугольник наибольшой площади, состоящий из одних нулей.

1 Описание

Требуется реализовать алгоритм динамического программирования, решающий поставленную задачу: найти наибольший прямоугольник, состоящий из нулей. Для решения этой задачи был использован метод гистограмм. В качестве высот столбцов гистограммы будут выступать количество подряд идущих нулей сверху у ячейки. Анализируем гистограмму: если столбцы неубывают по высоте, то продолжаем её построение, иначе фиксируем, какая наибольшая площадь прямоугольника могла быть получена до падения гистограммы, а затем снова продолжаем алгоритм. Подсчёт площади осуществляется так: запоминается количество стобцов в текущем неубывающем куске гистограммы и умножается на высоту наименьшего из столбцов. Из всех таких площадей выбирается наибольшая и выводится в качестве ответа.

2 Исходный код

В коде используется алгоритм мемоизации: смысла хранить всю матрицу нет, поэтому хранятся только текущая и предыдущая её строки. В стэке содержатся высоты столбцов гистограммы и индексы, на которых находятся эти стобцы. Когда высота верхнего элемента стека меньше текущего, то мы начинаем доставать все элементы до тех пор, пока не найдём элемент меньше или равный по высоте текущему. В процессе "вытаскивания" элементов подсчитывается промежуточная площадь: из текущего индекса вычитается индекс вытащенного элемента и это умножается на высоту вытащенного элемента. Значение в переменной *ans* обновляется, если оно меньше получившегося результата. Когда же высота верхнего элемента стека не меньше высоты текущего элемента, он добавляется в верхушку стека. Алгоритм продолжается до тех пор, пока не будут пройдены все строки матрицы.

```
#include <iostream>
#include <vector>
#include <stack>

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr); cout.tie(nullptr);
    int n, m;
    cin >> n >> m;
    int ans = 0;
    vector<int> matrix1(m);
    vector<int> matrix2(m);
    char a;
    for (int i = 0;i < n;i++) {
        fill(matrix2.begin(), matrix2.end(), 0);
        stack<pair<int, int>> s;
        for (int j = 0;j < m;j++) {
            cin >> a;
            if (a - '0' == 1) {
                continue;
            }
            else if (i == 0) {
                matrix2[j] = 1;
            }
            else { //
                matrix2[j] = matrix1[j] + 1;
            }
        }
        s.push({ 0, 0 }); //
        int high;
        int width;
```

```
34          int tmp;
35          for (int j = 0;j < m;j++) {
36              if (matrix2[j] > (s.top()).first) { //
37                  s.push({ matrix2[j], j });
38              }
39              else if (matrix2[j] < (s.top()).first) { //
40                  high = 1e3;
41                  width = 0;
42                  while (matrix2[j] < (s.top()).first) {
43                      high = min(high, (s.top()).first);
44                      width = j - (s.top()).second;
45                      tmp = (s.top()).second;
46                      ans = max(ans, high * width);
47                      s.pop();
48                  }
49                  if (matrix2[j] > (s.top()).first) {
50                      s.push({ matrix2[j], tmp });
51                  }
52              }
53              else { //
54                  continue;
55              }
56
57          }
58          high = 1e3;
59          width = 0;
60          while (!s.empty()) { //
61              high = min(high, (s.top()).first);
62              width = m - (s.top()).second;
63              ans = max(ans, high * width);
64              s.pop();
65          }
66          swap(matrix1, matrix2);
67      }
68      cout << ans;
69  }
```

3 Консоль

```
root@DESKTOP-5HM2HTK:~# cat <test
4 5
01011
10001
01000
11011
root@DESKTOP-5HM2HTK:~# g++ lab7.cpp
```

```
root@DESKTOP-5HM2HTK:~# ./a.out <test
4
root@DESKTOP-5HM2HTK:~#
```

4 Тест производительности

```
root@DESKTOP-5HM2HTK:~# g++ generator.cpp
root@DESKTOP-5HM2HTK:~# ./a.out tests
root@DESKTOP-5HM2HTK:~# g++ benchmark.cpp
root@DESKTOP-5HM2HTK:~# ./a.out <tests/1.t
Size of field is: 1000 X 1000
DP algorithm time: 273ms
Simple algoritm time: 1020ms
root@DESKTOP-5HM2HTK:~#
```

Как видно, алгоритм динамического программирования работает почти в 4 раза быстрее, чем наивный. Это неудивительно, ведь алгоритм на гистограммах проходит всё поле ровно один раз, а не несколько, как происходит в простом алгоритме

5 Выводы

Выполнив седьмую лабораторную работу по курсу «Дискретный анализ», я ознакомился с новым методом решения задач - метод гистограмм. Также я потренировался решать задачи методом динамического программирования.

Список литературы

[1] *Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ, 2-е издание. — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.)).*

[2] *Гистонрамма - Википедия.*
URL: `https://en.wikipedia.org/wiki/HistogramScott's_normal_reference_rule`
(дата обращения: 22.10.2022).