Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Дискретный анализ»

Студент: А. А. Терво
Преподаватель: С. А. Сорокин
Группа: М8О-307Б

Дата:
Оценка:
Подпись:

Москва, 2022

Курсовой проект

Задача: Вам дан набор горизонтальных отрезков, и набор точек, для каждой точки определите сколько отрезков лежит строго над ней.

В первой строке вам даны два числа n и m — количество отрезков и количество точек соответственно. В следующих n строках вам заданы отрезки, в виде троек чисел l, r и h — координаты x левой и правой границ отрезка и координата y отрезка соответственно. В следующих m строках вам даны пары чисел — координаты точек.

Для каждой точки выведите количество отрезков над ней.

Ваше решение должно работать *online*, то есть должно обрабатывать запросы по одному после построения необходимой структуры данных по входным данным. Чтение входных данных и запросов вместе и построение по ним общей структуры запрещено.

1 Описание

Требуется реализовать поиск количество отрезков, лежащих строго над точкой, используя персистентные структуры данных.

Задачу я буду решать при помощи алгоритма заметающей прямой.

Суть решения заключается в следующем: координатная плоскость разбивается на вертикальные полосы таким образом, чтобы каждый отрезок либо пересекал полосу полностью, либо не пересекал её вообще; таким образом, каждая полоса будет отличаться от предыдущей (левее неё) на один отрезок (в новой полосе либо начинается новый отрезок, либо кончается старый); для каждой полосы хранятся отрезки, пересекающие её в отсортированном виде; при осуществлении запроса сначала производится поиск полосы методом бинарного поиска (сложность $O(logn)$, где n — количество полос), а затем осуществляется поиск в сортированной структуре координаты y точки, что также выполняется за $O(logn)$; и, наконец, подсчитывается количество искомых отрезков.

В качестве отсортированной структуры буду использовать бинарное дерево, но сделаю его персистентным. При добавлении или удалении вершины будет создаваться новый корень, по которому можно будет получить доступ к нужной версии дерева.

Персистентность реализуется следующим образом: при добавлении или удалении вершины создаётся новая ветвь дерева до этой вершины, остальные ветви остаются нетронутыми, в новое дерево переносятся только указатели на их узлы.

2 Исходный код

Опишу класс $TNode$ узла дерева и класс $TTree$ самого дерева.

В классе дерева корни каждого состояния хранятся в векторе. Реализованы функции добавления и удаления узла, отладочные методы печати дерева на экран, проверки его на корректность добавления и удаления узлов, а также вспомогательные методы для поиска и печати.

```
  struct TNode {
    TNode* Left;
    TNode* Right;
    double Height;
    TNode(double h) {
      Height = h;
      Left = nullptr;
      Right = nullptr;
    }
    ~TNode() {}
  };

  class TTree {
  public:
    std::vector<TNode*> Roots;
    TTree() {}
    ~TTree() {}

    void PrintTree(int);
    TNode* AddNode(double);
    TNode* RemNode(double);
    bool Check(double);
    bool Check(double, TNode*);
    void FullCheck();
    void FullCheck(TNode*);
    std::vector<double> Values;

  private:
    bool RecFind(TNode*, double &);
    void RecPrintTree(TNode*, int);
  };

}
```

3 Консоль

```
[alext@alext-pc solution]$ make
g++ -std=c++14 -pedantic -Wall -Wextra-Wno-unused-variable lab1.cpp -o solution
[alext@alext-pc solution]$ cat test1
472891 asakdhfl
130391 bfsadfkjlsdf
891767 csdafKHdf
130000 dkhjs32
[alext@alext-pc solution]$ ./solution < test1
130000 dkhjs32
130391 bfsadfkjlsdf
472891 asakdhfl
891767 csdafKHdf
```

4 Тест производительности

Тут Вы описываете собственно тест производительности, сравнение Вашей реализации с уже существующими и т.д.

Тест производительности представляет из себя следующее: поиск образцов с помощью суффиксного массива сравнивается с поиском алгоритма КМП, но время на построение суффиксного массива не учитывается. Текст состоит из 1 миллиона букв: а образцов около 200 штук, длина которых может быть от 2 до 100 букв.

```
Andys-MacBook-Pro:kmp Andy$ g++ main.cpp
Andys-MacBook-Pro:kmp Andy$ ./a.out <../in.txt >out2.txt
Andys-MacBook-Pro:kmp Andy$ cat out2.txt | grep "time"
KMP search time: 1.639993 sec
Andys-MacBook-Pro:sa Andy$ make
g++ -pedantic -Wall -Werror -Wno-sign-compare -Wno-long-long -lm -O2 -o lab5
main.cpp suffix_tree.cpp suffix_array.cpp
Andys-MacBook-Pro:sa Andy$ ./lab5 <../in.txt >out1.txt
Andys-MacBook-Pro:sa Andy$ cat out1.txt | grep "time"
Suffix array build time: 2.179744 sec
Suffix array search time: 0.003511 sec
```

Как видно, что суффиксный массив выиграл у КМП, так как и т.д.

5 Выводы

Здесь Вы пишите то, чему научились на лабораторной на самом деле, что узнали нового, где это может пригодиться и т.д. Мне важно, какие именно Вы сделали выводы из лабораторной.

Выполнив первую лабораторную работу по курсу «Дискретный анализ», я научился тому-то и тому-то.

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] *Сортировка подсчётом — Викиконспекты.*
URL: `https://neerc.ifmo.ru/wiki/index.php?title=Сортировка_подсчётом` (дата обращения: 09.10.2020).

[3] Список использованных источников оформлять нужно по ГОСТ Р 7.05-2008