Задание 11 исправленное) Определить АТД Разреженная матрица, обеспечивающий метод get(i,ji,j) для получения элемента матрицы и set(i,j,vi,j,v) для изменения (добавления) ненулевого элемента. В конструкторе задаются размеры матрицы.
Реализовать АТД через словарь по ключам map<pair<int,int>, double>. Определить эффективность операций + и * в зависимости от количества ненулевых элементов KK.

Файл Правка Поиск Вид Действия Отладка Тестирование Изменения Настройки Подсветка Вкладки Справка

1 prac.cpp *

```cpp
#include <iostream>
#include <map>
using namespace std;
class SparseMatrix
{
private:
  int rows; // Количество строк
  int cols; // Количество столбцов
  map<pair<int, int>, double> elements; // Словарь для хранения ненулевых элементов

public:

  SparseMatrix(int numRows, int numCols) : rows(numRows), cols(numCols) {} // Конструктор с заданием размеров матрицы

  void set(int i, int j, double v)// Метод для установки значения элемента
  {
  if (i < 0 || i >= rows || j < 0 || j >= cols){
    throw out_of_range("Out of bounds");
  }

  elements[{i, j}] = v;
  }

  double get(int i, int j) const// Метод для получения значения элемента
{

  if (i < 0 || i >= rows || j < 0 || j >= cols){
    throw out_of_range("Out of bounds");
  }

  auto it = elements.find({ i, j });// Поиск значения элемента в словаре
  if (it != elements.end()){
    return it->second;
  }
  else{
    return 0;
  }
}
};
```

K1 – количество ненулевых элементов в 1 матрице

K2 – количество ненулевых элементов во второй матрице

M – количество строк во второй матрице

Операция +: O(K1+K2)

Операция *:O(K1*K2/M)

Задание 23 исправленное) Определите необходимые геометрические объекты и напишите следующую функцию
В декартовой системе координат на плоскости заданы координаты вершин треугольника и ещё одной точки. Определить, принадлежит ли эта точка треугольнику.
Для точки использовать класс из лекций и его методы.

1 prac.cpp *

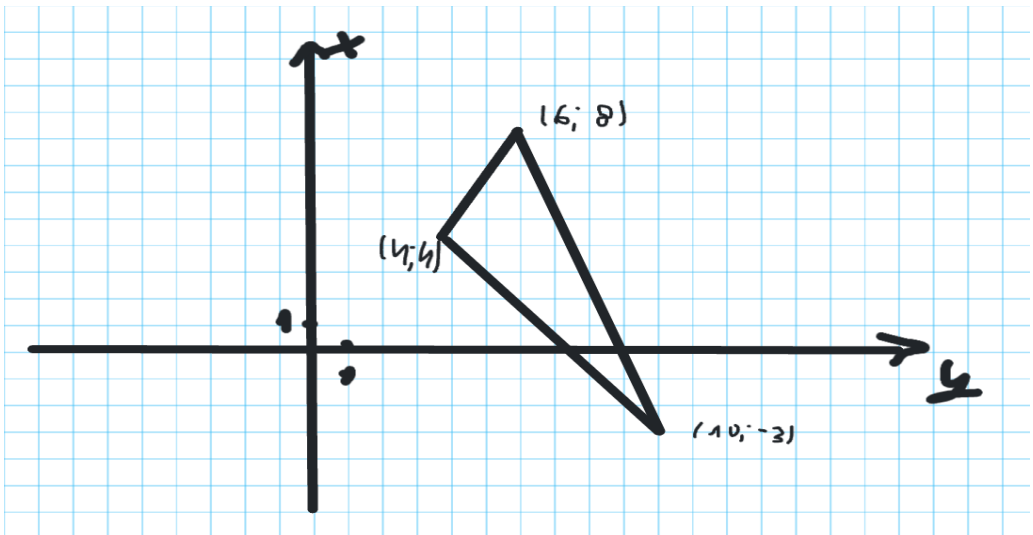
```cpp
#include <iostream>
#include <cmath>
using namespace std;
struct Point {
    double x, y;
    double len() const { return hypot(x, y); } // расстояние от начала координат
    double phi() const { return atan2(y, x); } // угол
    Point operator+(Point p) const { return {x + p.x, y + p.y}; }
    Point operator-(Point p) const { return {x - p.x, y - p.y}; }
    double operator*(Point p) const { return x * p.x + y * p.y; } // скалярное произведение
    double operator^(Point p) const { return x * p.y - y * p.x; } // векторное произведение
    Point operator*(double a) const { return {a * x, a * y}; } // "масштабирование"
    Point turn(double a) const { double ca = cos(a), sa = sin(a); return {x * ca - y * sa, -x * sa + y * ca}; } // поворот
    Point turn() const { return {-y, x}; } // поворот на π/2
    Point operator-() const { return {-x, -y}; } // поворот на π
};


bool isPointInTriangle(Point A, Point B, Point C, Point P) {
    // Векторные произведения
    double d1 = (B - A) ^ (P - A);
    double d2 = (C - B) ^ (P - B);
    double d3 = (A - C) ^ (P - C);
    bool below = (d1 < 0) || (d2 < 0) || (d3 < 0);
    bool upper = (d1 > 0) || (d2 > 0) || (d3 > 0);
    return !(below && upper);
}

int main() {
    Point A = {4, 4};//первая координата треугольника
    Point B = {6, 8};//вторая координата треугольника
    Point C = {10, -3};//третья координата треугольника
    Point P = {2, 4};// проверяемая точка

    if (isPointInTriangle(A, B, C, P)) {
        cout << "Точка P принадлежит треугольнику ABC.\n" ;
    } else {
        cout << "Точка P не принадлежит треугольнику ABC.\n";
    }

    return 0;
}
```



Задание 3 исправленное) Определите АТД для хранения информации о таблице базы данных: столбцы (количество и названия столбцов), строки со значениями (все значения в строке таблицы имеют тип string), ключи для получения строк таблицы в некотором порядке (для упрощения ключ включает только один столбец, т.е. ключ - это имя или номер столбца, а строки таблицы можно получать последовательно по одной в порядке

возрастания значения в указанном столбце). Перечислите методы АТД, обеспечивающие последовательный доступ к информации и её изменение, аргументы и возвращаемые значения каждого метода с комментариями.

Методы АТД:

CreateTable(CountStolb:int,StolbName:array<string>)->None

Описание: Создает новую пустую таблицу с заданным количеством и названиями столбцов.

- Аргументы:

CountStolb - количество столбцов в таблице.

StolbName - массив названий столбцов.

- Возвращаемое значение: None

Stradd(Values:array<string>) -> None

Описание: Добавляет новую строку с заданными значениями в таблицу.

-Аргументы:

Values - массив значений для новой строки. Длина массива должна совпадать с количеством столбцов в таблице.

- Возвращаемое значение: None

StrDel(index:int)->None

Описание: Удаляет строку по указанному индексу

-Аргументы:

Index – индекс строки, которую надо получить

-Возвращаемое значение: None

StrGet(index:int)->array<string>

Описание: Возвращает значение строки по индексу

-Аргументы:

Index – индекс строки, которую надо получить

-Возвращаемое значение: Массив значений строки

ValueUpd(indexstr:int,indexstolb:int,NewValue:string) ->None

Описание: Обновляет значение в указанной ячейке.

- Аргументы:

- indexstr - индекс строки, в которой нужно обновить значение.

- индексstolb - индекс столбца, в котором нужно обновить значение.

- NewValue - новое значение для ячейки.

- Возвращаемое значение: None

SetupSortKey(StolbName:string)-> None

Описание: Устанавливает столбец, по которому строки будут сортироваться.

- Аргументы:

StolbName - название столбца, по которому будет осуществляться сортировка.

- Возвращаемое значение: None

GetNextStr()->array<string>

Описание: Возвращает следующую строку в отсортированном порядке по ключевому столбцу. Если достигнут конец таблицы, возвращает пустой массив или специальное значение.

- Аргументы:None

- Возвращаемое значение: Массив значений следующей строки в порядке возрастания значений ключевого столбца.

GetColumnNames() -> array<string>

Описание: Возвращает массив имен всех столбцов таблицы.

- Аргументы: Нет

- Возвращаемое значение: Массив строк с именами столбцов.

GetColumnIndex(StolbName: string) -> int

Описание: Возвращает индекс столбца по его имени.

- Аргументы:

- StolbName - имя столбца, индекс которого нужно получить.

- Возвращаемое значение: Индекс столбца.

Задание 4 исправленное) Предложите структуры данных для представления АТД из задания 3. Перечислите поля, их типы и комментарии к каждому полю. Укажите оценку эффективности (амортизированную или среднюю) для каждого метода с учетом использованных структур данных. Хранимая в структуре информация не должна дублироваться.

Структуры данных и их поля:

Таблица

StolbName: array<string>— Массив названий столбцов

Stroki: array<array<string>> — Массив массивов строк, где каждая строка представляет собой массив значений.

StolbKey: string — Название столбца, по которому осуществляется сортировка.

indexStolbKey: int — Индекс ключевого столбца.

SortedIndex: array<int> — Отсортированный массив индексов строк для итерации по ключевому столбцу.

Iterator: int — Текущий индекс для итерации по отсортированным индексам.

Оценка эффективности методов:

CreateTable:O(1)

Stradd:O(n)

StrDel:O(n)

StrGet:O(1)

ValueUpd:O(1)

SetupSortKey:O(nlogn)

GetNextStr:O(1)

ResetIterator:O(1)

Задание 16) Есть nn ($2 \le n \le 100000$ $2 \le n \le 10000$) городов, заданных своими координатами (x_i, y_i)(xi, yi), i ∈ 1... n$i \in 1 \ldots n$, а также mm ($2 \le m \le 1000000$ $2 \le m \le 100000$) дорог, соединяющих города. Нужно построить дополнительное число дорог (возможно, нулевое), так чтобы из любого города можно было доехать в любой другой, двигаясь по дорогам. При этом сумма длин построенных дорог должна быть

минимально возможной.
Укажите какой алгоритм построения минимального остовного дерева является более эффективным для решения этой задачи и обоснуйте свой выбор.
Для решения этой задачи не нужно писать код, нужно:
1) указать количество вершин и ребер в графе
2) написать оценки эффективности O(...) для сравниваемых алгоритмов
3) сравнить эти оценки для верхних ограничений и указать меньшую
Это называется в математике "обоснованием выбора".

1) Количество вершин : $2 \leq n \leq 10000$)
   Количество ребер: ($2 \leq m \leq 100000$)
2) Для решения задачи рассмотрим два основных алгоритма: алгоритм Прима и алгоритм Крускала
   Алгоритм Прима обычно реализуется с использованием приоритетной очереди, что дает сложность O((n+m)logn)
   Алгоритм Крускала зачастую реализуется с помощью структуры данных Union-Find(объединение-поиск), что дает сложность O(mlogm)
3) Верхние ограничения n = 10000, m = 100000
   Алгоритм Прима:
   O(10000+100000)log10000) = O(440000)
   Алгоритм Крускала:
   O(100000 log 100000) = O(50000)

Исходя из подставленных вверхних ограничений, количество операций у алгоритма Прима на 60000 меньше. Таким образом, алгоритм Прима имеет меньшую ассимптотическую сложность,следовательно, будет являться более эффективным для решения данной задачи

Задание 17) Укажите какой алгоритм нужно использовать для решения задачи и обоснуйте свой выбор. Из каких вершин и ребер граф будет состоять?
Джон решил украсить дом иллюминацией к Рождеству. Он соединил проводами несколько ламп и подключил их к электросети. Затем Джон повернул рубильник, но некоторые лампы не загорелись. Напишите программу, которая определит, используя

информацию о выполненных соединениях, какие лампы будут гореть.
В первой строке ввода содержатся два целых числа, разделенных пробелом – количество ламп NN (1≤N≤1001≤N≤100) и количество проводов KK (1≤K≤10001≤K≤1000). Далее следует KK строк, в каждой строке сначала указывается номер элемента (0 для электросети, от 1 до NN для лампы), к которому был присоединен один из концов провода, потом через пробел – номер контакта 1 или 2 (все элементы имеют два контакта), затем указывается элемент и контакт, к которому был присоединен другой конец провода, в том же формате.
В выходной файл в первой строке вывести NN целых чисел, разделяя их пробелами – если ii-ая лампа будет гореть, то ii-ое число должно быть равно 1, иначе 0. Лампа будет гореть, только если ее контакты подсоединены к вершинам с различными потенциалами. Провод имеет нулевое сопротивление, поэтому разность потенциалов на вершинах, соединенных между собой проводами без нагрузки в виде других ламп будет равна 0.

Для решения данной задачи следует воспользоваться алгоритмом поиска компонент связности в графе. Поскольку лампы будут гореть только когда их контакты подсоединены к вершинам с различными потенциалами. То есть нужно проверить, принадлежат ли контакты одной лампы к разным компонентам. Так же в качестве аргумента за выбор этого алгоритма можно взять его эффективность, так как алгоритмы поиска компонент связности работают за линейное время O(Количество вершин+Количество ребер).

1) Вершины:
   Каждая лампа имеет по два контакта, то есть, для каждой лампы будет две вершины. Вершина для электросети(нулевая вершина)
   Следовательно, общее количество вершин будет 2N+1, где N- количество ламп.
2) Ребра:
   Ребра представляют собой провода, соединяющие контакты
   Если провод соединяется контакт а1 лампы б1 с контактом а2 лампы б2, то в графе будет ребро между соответствующими вершинами. Если провод соединяет контакт с электросетью, то будет ребро между соответствующей вершиной и нулевой вершиной

Задание 10) Определить АТД Полином, обеспечивающий метод calc для вычисления значения полинома в точке xx (используйте схему Горнера или барицентрическую форму интерполяционного многочлена Лагранжа). Реализовать полином через представление на значениях в точках. В конструкторе задается набор значений y0,...,yn−1y0,...,yn-1, x0x0 и ΔxΔx (xi=x0+i·Δxxi=x0+i·Δx). Определить операцию +

1 prac.cpp *

```cpp
#include <iostream>
#include <vector>
using namespace std;
class Polynom
{
private:
 vector<double> y;
 double x0;
 double deltaX;
public:
 // Конструктор с заданием значений y, начальной точки x0 и шага delta x
 Polynom(const vector<double>& values, double x0, double deltaX)
 : y(values), x0(x0), deltaX(deltaX) {}
 // Метод для вычисления значения полинома в точке x
 double calc(double x) const
 {
 int n = y.size();
 if (n == 0)
 {
 throw std::runtime_error("Polynom has no values");
 }
 double result = 0.0;
 for (int i = 0; i < n; i++)
 {
 double term = y[i];
 for (int j = 0; j < n; j++)
 {
 if (i != j)
 {
 term *= (x - (x0 + j * deltaX)) / ((x0 + i * deltaX) - (x0 + j *
deltaX));
 }
 }
 result += term;
 }
 return result;
 }
 // Операция сложения двух полиномов
 Polynom operator+(const Polynom& other) const
 {
 if (deltaX != other.deltaX || x0 != other.x0 || y.size() != other.y.size())
 {
 throw invalid_argument("Polynomials must have the same deltaX, x0, and size");
 }
 vector<double> newY(y.size());
 for (size_t i = 0; i < y.size(); i++)
 {
 newY[i] = y[i] + other.y[i];
 }
 return Polynom(newY, x0, deltaX);
 }
};
```

Задание 25) Напишите функцию бинарного возведения в степень по модулю M.

```cpp
#include <iostream>

// Функция бинарного возведения в степень по модулю M
long long mod_exp(long long base, long long exponent, long long mod) {
    long long result = 1;
    base = base % mod;  //base<mod

    while (exponent > 0) {
        if (exponent % 2 == 1) {
            result = (result * base) % mod;
        }

        base = (base * base) % mod;

        // Переходим к следующему биту экспоненты
        exponent = exponent / 2;
    }

    return result;
}
```