Звіт з лабораторної роботи №4
на тему «Регіональний пошук: метод 2-d дерева»
з дисципліни «Комп'ютерна графіка»
студентки 3-го курсу Факультету комп'ютерних наук та кібернетики
групи ІПС-32
Бондарець Дарини Володимирівни

Постановка задачі.
На площині (у просторі R^2) задано множину точок S.
Запитний регіон визначений як прямокутник та заданий своїми двома протилежними вершинами, лівою нижньою L та правою верхньою R вершинам.
Необхідно визначити, які з точок множини S лежать у прямокутній області $\{(x;y) \mid x(L)<x<(R), y(L) < y < y(R)\}$, при чому для запитного регіону допускаються випадки $x_{i=}$ INF та y_i=INF.

Розв'язання.
Роз'вязання задачі полягає у рекурсивному розбитті площини на прямокутні області, та відповідній йому побудові 2-D-дерева. Побудова відбувається одноразово. Після цього можна робити масові запити, тобто задавати регіон пошуку (прямокутник), та за обходом дерева визначати точки, що належать регіону.

Отже, розглянемо алгоритм виконання програми, яка дає відповідь на поставлену задачу.
На вхід програми задаються точки запитної множини (теоретично необмежена кількість, фактично, вважаємо, що N) та 2 точки запитного регіону (протилежні вершини прямокутника, нехай L та R).
Вузол дерева визначений як структура, що містить поточну вершину, задану своїми координатами $(x,y)_i$ – P(v), R(v) – прямокутну область, S(v) – множину точок, яка належить R(v), I(v) – розрізаючу пряму, яка проходить через точку P(v) та D – запитний регіон.
Побудуємо 2-Д-дерево. Упорядкуємо точки (відсортуємо) вхідної множини за зростанням координати «x». Знаходимо медіану (середню за впорядкуванням точку) нашої множини та проводимо вертикальну пряму через неї. Це корінь дерева. Надалі отримали дві підмножини вхідної множини (умовно ліву та праву). Розглянемо «ліву» підмножину. Відсортуємо точки на цей раз за зростанням ординати. Визначаємо медіану та проводимо горизонтальну пряму через неї. Це є лівий син кореня. Надалі для кожної з отриманих таким розбиттям підмножин запускаємо рекурсивно цей же алгоритм, допоки, не отримаємо розбиття площини на узагальнені прямокутники, які не мають всередині точок. Важливо на кожному кроці алгоритму зберігати у вузлах дерева вказані вище параметри. Таким чином отримаємо побудоване двійкове дерево пошуку.

Алгоритм пошуку
Алгоритм є рекурсивним (обхід дерева). Його необхідно виконати для кореня дерева, як вхідного даного.

Пошук()

1) якщо пустий перетин D та R1(v) && непустий перетин D та R2(v), то лівий пошук (в перетині D та R1(v))
2) якщо пустий перетин D та R2(v) && непустий перетин D та R2(v), то правий пошук (в перетині D та R2(v))
3) якщо непустий перетин D та R1(v) && непустий перетин D та R2(v), то лівий пошук (в перетині D та R1(v)) і правий пошук (в перетині D та R2(v))
4) перевірка P(v) не належить D.

Відповідно зберігаємо дані перевірок і після виконання алгоритму отримаємо список верши, що містяться і регіоні пошуку.

Тут R(v) – прямокутна область, у якій міститься вершина v, R(v)= R1(v)+R2(v), тобто R1 та R2 – це області, отримані в результаті розрізу площини прямою, що проходить через v.

Оцінка складності:

Пам’ять O(N).
Побудова дерева – O(NlogN).
Найгірший випадок виконання алгоритму – O(sqrt(N)).

Мова реалізації: Java.