Грищенко Юрій ІПС-32
Лабораторна робота №7

Постановка задачі: Підтримка оболонки. Задані спочатку порожня множина S та послідовність із N точок, кожна з яких або додається до множини S, або вилучається з неї (очевидно, за умови, що вона належить S). Необхідно підтримувати опуклу оболонку множини S.

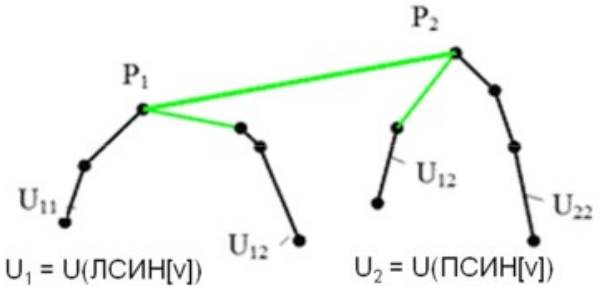
Опис алгоритму (ідея алгоритму — Overmars та van Leeuwen)**:**

Буде використаний той факт, що границя опуклої оболонки є об'єднанням двох (опуклих) монотонних ламаних ліній (ланцюгів), які обмежують оболонку зверху і знизу. Розглянемо побудову верхньої оболонки.

Основа – збалансоване за висотою двійкове дерево пошуку, листки якого використовуються для збереження точок поточної множини. Кожен проміжний вузол представляє верхню оболонку точок, що зберігаються в листках відповідного піддерева.

Процедура пошуку буде проводитися відповідно до значення абсциси точок, тобто проходження листків дерева зліва направо дає множину точок, впорядковану за x-координатою.

Використовується функція З’ЄДНАТИ(U_1, U_2) яка дозволяє знайти опорний відрізок для двох оболонок U_1, U_2:



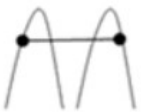
U зберігаються як зчеплені черги.

Схема функції З’ЄДНАТИ:
1. Знайти p_1 і p_2
2. Розчепити U_1 на U_{11} та U_{12}.
2. Розчепити U_2 на U_{21} та U_{22}.
4. Зчепити U_{11} та U_{22}.

Кожна з цих операцій займає час O(log n) для зчеплених черг.

Пошук точок p_1 та p_2:

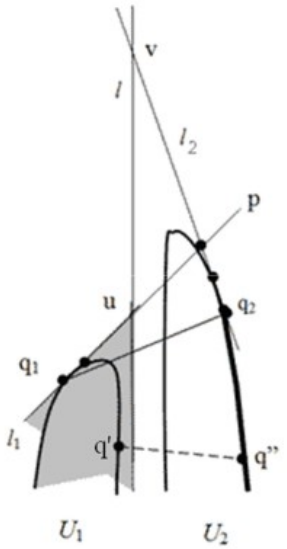
Зчеплена черга має вигляд збалансованого дерева пошуку — починаємо з точкок-коренів, маємо q_1, q_2. Кожна із цих двох вершин може бути класифікована відносно відрізка $[q_1, q_2]$ як опукла, опорна або ввігнута. Залежно від класифікації можливі 9 випадків.

$q_1 \mid q_2$	ввігнута	опорна	опукла
ввігнута			
опорна			
опукла			

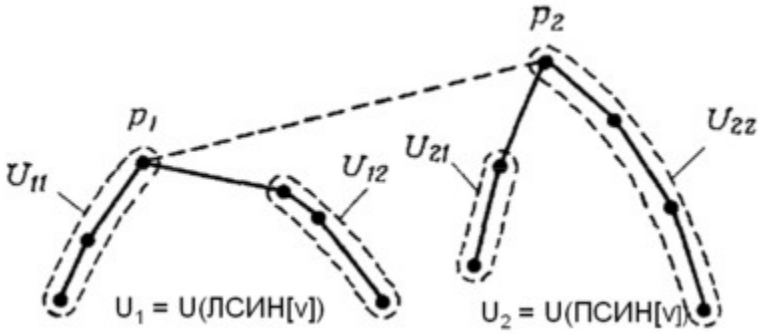
1.	$v(q_1)$ (або ПС $[v(q_1)]$)	ЛС $[v(q_2)]$ (або $v(q_2)$)
2.	ПС $[v(q_1)]$	ПС $[v(q_2)]$
3.	$v(q_1)$	ПС $[v(q_2)]$
4.	ЛС $[v(q_1)]$	ЛС $[v(q_2)]$
5.	Результат	Результат
6.	ЛС $[v(q_1)]$	ПС $[v(q_2)]$
7.	ЛС $[v(q_1)]$	$v(q_2)$
8.	ЛС $[v(q_1)]$	ПС $[v(q_2)]$
9.	ЛС $[v(q_1)]$	ПС $[v(q_2)]$

Більш детально розглянемо випадок 1:

Проведемо прямі l_1 (від q_1 до правого сусіда) та l_2 (від q_2 до лівого сусіда). Якщо їх точка перетину p знаходится справа від вертикалі, що відділяє дві оболонки (як на малюнку), то можна відкинути точки, правіші за q_2. Інакше відкидаємо точки, лівіші за q_1.



Обернена операція: маючи U(v) знайти U(ЛСИН(v)) та U(ПСИН(v)).

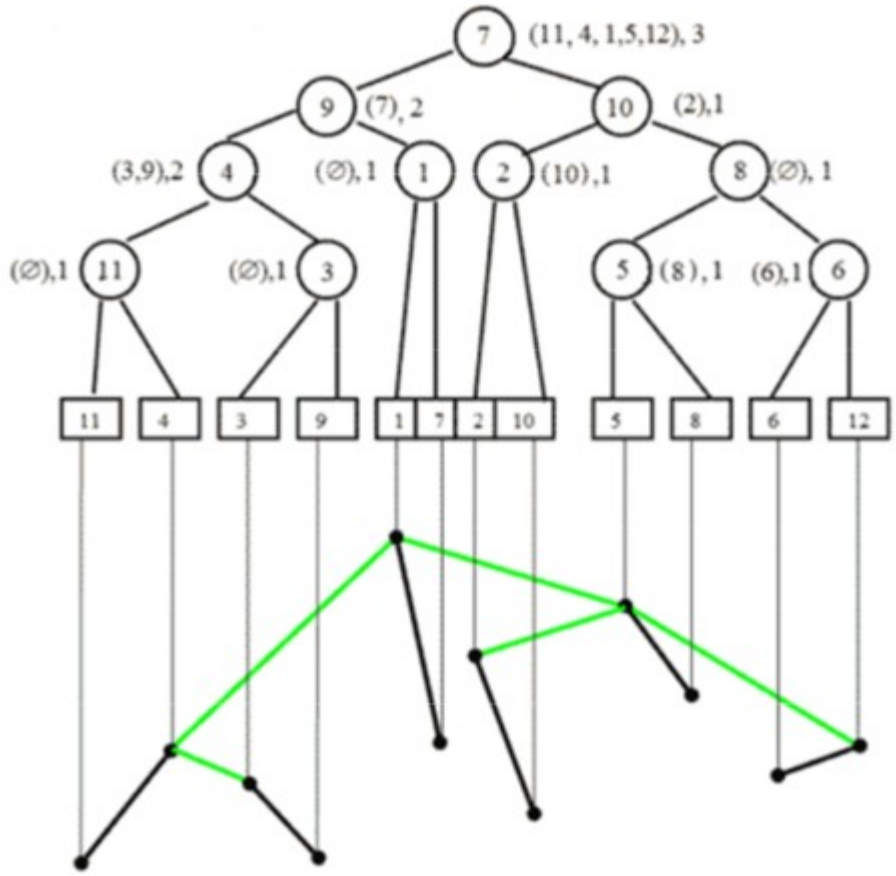


Знаючи ребро $[p_1, p_2]$ (яке визначається індексом точки p_1 у U(v)), можна розчепити U(v) на U_{11} та U_{22}, та зчепити їх з ланцюгами, які зберігаються у ЛСИН(v) та ПСИН(v) відповідно.

Отже, у вузлах v дерева також зберігаються:

- Індекс J(v) лівої опорної точки в U(v)
- Вказівник на зчеплену чергу Q(v), яка містить частину U(v), що не входить до U(БАТЬКО(v)) (якщо v — корінь, то Q(v) = U(v))

Приклад дерева:



Вставка точки p: спуск, вставка, підйом.

procedure СПУСК(v, p)
  begin if $(v \neq$ лист$)$ **then**
    begin
      $(Q_L, Q_R) :=$ РОЗЧЕПИТИ$(U[v], J[v])$;
      $U[$ЛСИН$[v]] :=$ ЗЧЕПИТИ$(Q_L, Q[$ЛСИН$[v]])$;
      $[$ПСИН$[v]] :=$ ЗЧЕПИТИ$(Q[$ПСИН$[v]], Q_R R)$;
      if $(x[p] \leq x[v])$ **then**
        $v :=$ ЛСИН$[v]$
      else $v :=$ ПСИН$[v]$;
      СПУСК(v, p);
    end;
  end.

(знаючи U(v) батька, знаходимо опуклі оболонки синів)

Далі вставляється точка в лист: для простоти вважатимемо, що дерево балансувати не потрібно (це збільшість кількість вузлів, які потрібно обробити, не приводячи до принципових відмінностей).

```
procedure ПІДЙОМ(v)
  begin if (v ≠ корінь) then
    begin
      (Q1, Q2, Q3, Q4, J) := З'ЄДНАТИ(U[v], U[БРАТ[v]]);
      Q[ЛСИН[БАТЬКО[v]]] := Q2;
      Q[ПСИН[БАТЬКО[v]]] := Q3;
      U[БАТЬКО[v]] := ЗЧЕПИТИ(Q1, Q4);
      J[БАТЬКО[v]] := J;
      ПІДЙОМ(БАТЬКО[v]);
    end;
  else Q[v]: = U[v];
end.
```

(оновлюємо атрибути вузлів: знаючи опуклі оболонки синів, знаходимо U(БАТЬКО(v))

Складність алгоритму:

Кожен крок СПУСКу та ПІДЙОМу відбувається за час O(log n), висота збалансованого дерева log n, отже часові витрати на вставку дорівнюють $O(\log^2 n)$ у найгіршому випадку.

Це достатньо швидко, щоб обробляти дані в реальному часі, але повільніше за алгоритм Препарати. Алгоритм Препарати має затриму O(log n) для вставки елементів, що є теоретичним мінімумом для відкритих алгоритмів.

Перевага даного алгоритму полягає в тому, що він зберігає інформацію про всі точки, і внутрішні також, тому він дозволяє не тільки додавати точки в множину, а і видаляти.

Реалізовано на мові Python.

Інтерфейс користувача: набір точок задається файлом points.txt.
Програма додає точки в множину S одна за одною, і кожен раз у вікні показує опуклу оболонку (як і верхній, так і нижній ланцюг). У консоль виводяться дерева для обох ланцюгів та значення U(v) після спуску і підйому (як debug-інформація).