Лабораторна робота №6

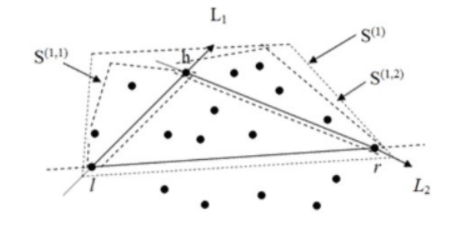
Снєговського Владислава, ІПС-31

Завдання:

Швидкий метод побудови опуклої оболонки

Теорія:

Початкове розбиття множини визначається прямою, яка проходить через дві точки l і r, з найменшою і найбільшою абсцисами відповідно.



S1 – підмножина точок, що розташовані вище або на прямій, яка проходить через l і r і S2 – симетричним чином визначена підмножина точок, розташованих нижче або на тій самій прямій . Далі розглядаємо будь-яку з підмножин (наприклад S1), визначимо для неї точку h для якої трикутник hlr має максимальну площу серед усіх трикутників. Якщо таких точок декілька, то обираємо лівішу. Рекурсивно обходимо всі підмножини утворюючи оболонку.

Час: N log N, у найгіршому випадку N2

Складність така ж сама як і у QuickSort

Головна функція з коду:

*def* \_\_add\_farthest(self, *points*, *p1*, *p2*, *side*, *result*):  
 farthest = *None* max\_area2 = 0  
  
 # Check if any point is farther than p1, p2  
 *for* p *in points*:  
 *if* self.\_\_get\_side(*p1*, *p2*, p) == *side*:  
 area2 = abs(self.\_\_area2(*p1*, *p2*, p))  
 *if* area2 > max\_area2:  
 farthest = p  
 max\_area2 = area2  
  
 # If there is no farther point, stop recursivity and add p1 and p2 to the convex  
 # hull list if they aren't already in it  
 *if* farthest == *None*:  
 self.\_\_add\_extremes(*p1*, *p2*, *result*)  
 *return* # If there is, new recursivity calls checking for new farthest points on each side  
 self.\_\_add\_farthest(*points*, farthest, *p1*, - self.\_\_get\_side(farthest, *p1*, *p2*), *result*)  
 self.\_\_add\_farthest(*points*, farthest, *p2*, - self.\_\_get\_side(farthest, *p2*, *p1*), *result*)