Выпуклая оболочка.

Гащук Елизавета, 332 группа

1.Постановка задачи. Дан набор точек на плоскости. Требуется вычислить выпуклую оболочку данного множества точек.

2.1.Алгоритм.

Псевдокод:

Ввод: множество P точек на плоскости (запись координат точек в файл points.txt).

Вывод: массив, содержащий вершины выпуклой оболочки (запись вершин выпуклой оболочки в файл hull.txt).

1. отсортировать точки по координате x, получили поседовательность точек $p_0, ..., p_{n-1}$.
2. отсортировать подпоследовательности, образованные одинаковой координатой x, по y.
3. положить точки p_0 и p_1 в лист L_{upper}, сделав p_0 первой.
4. **for** i←2 **to** n - 1
5. **do** добавить p_i в L_{upper}.
6. **while** L_{upper} содержит более двух точек **and** последние три точки из L_{upper} не образуют правый поворот
7. **do** удалить среднюю точку среди последних трех из L_{upper}.
8. положить точки p_{n-1} and p_{n-2} в лист L_{lower}, сделав p_{n-1} первой.
9. **for** i←n - 3 **downto** 0
10. **do** добавить p_i to L_{lower}.
11. **while** L_{lower} содержит более двух точек **and** последние три точки из L_{lower} не образуют правый поворот
12. **do** удалить среднюю точку среди последних трех из L_{lower}.
13. удалить первую и последнюю точки в L_{lower}, так как они дублируются в L_{upper}
14. **вернуть** $L_{upper} + L_{lower}$ (запись вершин выпуклой оболочки в файл hull.txt)

2.2. Подалгоритмы:

а). сортировка подмассивов:

points = $p_0, ..., p_{n-1}$ - массив наших точек, отсортированных по координате x.

count = 0 - количество точек, имеющих одинаковую x - координату.

for i←1 **to** n - 1

if count == n - 1

do сортировать точки массива points, отсортированного по x - координте, начиная с той, что под номером 0, заканчивая с той, что под номером n - 1;

if $p_{i-1}(x) == p(x)$

increase count;

if i == n - 1

do сортировать точки массива points, отсортированного по x - координте, начиная с той, что под номером i - count, заканчивая с той, что под номером i;

else

do сортировать точки массива points, отсортированного по x - координте, начиная с той, что под номером i - 1 - count, заканчивая с той, что под номером i - 1;

count = 0;

б). "поворот правый или левый?":

Без ограничения общности будем считать, что точки лежат на плоскости z = 0.

Пусть $V = (v_x, v_y, 0)$, $U = (u_x, u_y, 0)$. Рассмотрим векторное произведение векторов $\vec{V}$ и $\vec{U}$. Если [$\vec{U}$, $\vec{V}$], то получим положительно ориентированную тройку, если [$\vec{V}$, $\vec{U}$] - иначе. Разберем случай [$\vec{U}$, $\vec{V}$]:

$$\boldsymbol{[\vec{U}, \vec{V}] = (\begin{vmatrix} u_y & 0 \\ v_y & 0 \end{vmatrix}, \begin{vmatrix} 0 & u_x \\ 0 & v_x \end{vmatrix}, \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix}) = (0, 0, \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix})}$$

$$Sign([\vec{U}, \vec{V}]) = 1. \text{ Распишем подробнее, что такое} \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix}:$$

$$\begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix} = \begin{vmatrix} \mathrm{C}_x - \mathrm{A_x} & \mathrm{C}_y - \mathrm{A_y} \\ \mathrm{B}_x - \mathrm{A_x} & \mathrm{B}_y - \mathrm{A_y} \end{vmatrix} = - \begin{vmatrix} 1 & A_x & A_y \\ 1 & \mathrm{B}_x & \mathrm{B}_y \\ 1 & C_x & C_y \end{vmatrix}.$$

Если взять отрицательно ориентированную тройку, расписать все то же самое для нее, то получим:

$$\boldsymbol{[\vec{V}, \vec{U}]_z = \begin{vmatrix} 1 & A_x & A_y \\ 1 & \mathrm{B}_x & \mathrm{B}_y \\ 1 & C_x & C_y \end{vmatrix}.}$$

Т.е. если взять третью координату векторного произведения и посмотреть на знак, то мы сможем отличить правый поворот от левого.

$$\text{Итого имеем}: \begin{cases} -1 & \text{при правом повороте;} \\ 1 & \text{при левом повороте;} \end{cases}$$

В программе данную проверку выполняет функция right turn.

в). сортировка : использую метод сортировки Хоара.

3.Сложность.

- Сортировка Хоара имеет сложность $n\log(n)$. Один раз сортируем по x координате $\rightarrow n\log(n)$.
- После сортировки по x - координате получили k подпоследовательностей, образованных точками с одинаковой x - координатой. Скажем, что длина i - ой подпоследовательности есть len_i. Тогда нам нужно еще $\sum_{i=1}^{k} len_i \log(len_i)$.
- Далее n - 2 раза проверяем поворот. Тут требуется $11(n-2)$ действий (для L_{upper}).
- Далее n - 2 раза проверяем поворот. Тут требуется $11(n-2)$ действий (для L_{lower}).

Итого имеем: $n\log(n) + \sum_{i=1}^{k} len_i \log(len_i) + 2{\cdot}11(n-2)$

4.Результаты.