به نام خدا

نام: فاطمه زهرا بخشنده **استاد: دکتر مهدی کریمی نسب**

شماره دانشجویی: 98522157 **درس مبانی نظریه گراف**

تمرین امتیازی حل مسئله TSP با استفاده از الگوریتم Nearest Neighbor:

8 نقطه زیر را به عنوان ورودی برنامه می دهیم:

```
inputs = [
    (2, 0),
    (0, 5),
    (1, 1),
    (6, 5),
    (0, 0),
    (1, 2),
    (6, 1),
    (3, 4)
]
```

کد پایتون اجرا شده و در مرحله اول رئوس روی پوسته محدب را به ترتیب می نویسد. سپس با استفاده از پوسته محدب، الگوریتم Nearest Neighbor را اجرا می کنیم.

به این صورت که هر دفعه یکی از نقاط درون پوسته محدب را در نظر گرفته، و مجموع فاصله آن از هر دو راس پشت هم در پوسته محدب را می‌سنجیم. هر کدام که کمترین فاصله را داشت این نقطه را بین آن دو راس قرار می دهیم. در آخر راس های پشت هم در tour پیدا شده در مسئله TSP را به ترتیب بر می‌گردانیم.

همچنین معادلات خطوط گذرنده از یال های این tour را نیز به ترتیب خروجی می دهیم.

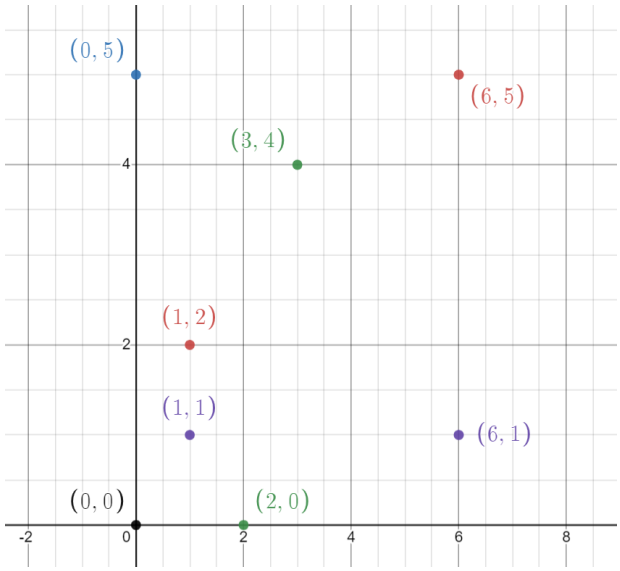
عکس خروجی در صفحه بعد قرار دارد.

```
1. founded a convex hull:
(0, 5)
(6, 5)
(6, 1)
(2, 0)
(0, 0)

2. founded a tour using nearest neighbor algorythm:
(0, 5)
(3, 4)
(6, 5)
(6, 1)
(2, 0)
(1, 2)
(1, 1)
(0, 0)
```

```
3. tour edges lines:
y = -0.33 x + 5.0
y = 0.33 x + 3.0
x =  6.0
y = 0.25 x - 0.5
y = -2.0 x + 4.0
x =  1.0
y = 1.0 x
x =  0.0
```

نقاط داده شده در ورودی به صورت زیر بودند. می بینیم که این کد (در خروجی که در صفحه قبل قرار دادم) به درستی یک tour را از آن ها پیدا کرده که در آن ترتیب قرارگیری رئوسی که در پوسته محدب بدست آمد نیز حفظ شده است.



کد پایتون نیز به شرح زیر است: (کد های پیدا کردن convex_hull و یکسری تابع های دیگر، در فایل graham.py پیاده سازی شده است، که در تمرین قبل تحویل داده شد. و در این کد از آن ها استفاده کرده ام)

```
from graham import *
import numpy as np
import math

def TSP(points: list):
    print('\n1. founded a convex hull:\n')
    points_on_convex_hull = find_convex_hull(points)
    points_in_convex_hull = list(set(points).difference(points_on_convex_hull))

    points_sorted = points_on_convex_hull.copy()
    for p_in in points_in_convex_hull:
        min_dist = math.inf
        min_i = None
        for i in range(-1, len(points_sorted) - 1):
            p_on_1, p_on_2 = points_sorted[i], points_sorted[i + 1]
            dist = distance(p_on_1, p_in) + distance(p_on_2, p_in)
            if dist < min_dist:
                min_i = i
```

```
                min_dist = dist
        if min_i == -1:
            points_sorted.append(p_in)
        else:
            points_sorted.insert(min_i + 1, p_in)

    return points_sorted


if __name__ == "__main__":

    inputs = [
        (2, 0),
        (0, 5),
        (1, 1),
        (6, 5),
        (0, 0),
        (1, 2),
        (6, 1),
        (3, 4)
    ]

    points = [Point(x, y) for (x, y) in inputs]

    tour = TSP(points)
    print('\n2. founded a tour using nearest neighbor algorythm:\n')
    for p in tour:
        print("(" + str(p.x) + ", " + str(p.y) + ")\n")

    print('\n3. tour edges lines:\n')
    for i in range(len(tour) - 1):
        find_line(tour[i], tour[i + 1])
    find_line(tour[-1], tour[0])
```