Tecnológico de Monterrey

Tecnológico de Monterrey - Campus Monterrey
School of Engineering and Sciences
Engineering in Computational Technologies
Analysis and Design of Advanced Algorithms

Class Activity 7:
Intersection of Segments & closest pair

Group: 607
Team #3
Luis Salomón Flores Ugalde

Santiago Quintana Moreno A01571222
Miguel Ángel Álvarez Hermida a01722925

File Edit Selection View Go Run Terminal Help

2.Advanced Algorithms

ClassAct7_IntersectionSegmentsClosestPair.py

ClassAct_7 > ClassAct7_IntersectionSegmentsClosestPair.py > dist3

```python
# Analysis and Design of Advanced Algorithms
# Group #607
# Team 3
# Luis Salomón Flores Ugalde

# Santiago Quintana Moreno A01571222
# Miguel Ángel Álvarez Hermida A01722925

# ------ Intersection of Segments & closest pair ------
from math import hypot
from typing import List, Tuple, Optional
import sys


Point2 = Tuple[float, float]
Point3 = Tuple[float, float, float]

def dist2(p: Point2, q: Point2) -> float:
    return hypot(p[0] - q[0], p[1] - q[1])

def dist3(p: Point3, q: Point3) -> float:
    dx, dy, dz = p[0] - q[0], p[1] - q[1], p[2] - q[2]
    return (dx*dx + dy*dy + dz*dz) ** 0.5

def closest_pair_bruteforce_2d(pts: List[Point2]):
    n = len(pts)
    best = float("inf")
    pair = (None, None)
    for i in range(n):
        for j in range(i+1, n):
            d = dist2(pts[i], pts[j])
            if d < best:
                best, pair = d, (pts[i], pts[j])
    return best, pair

def closest_pair_bruteforce_3d(pts: List[Point3]):
    n = len(pts)
    best = float("inf")
    pair = (None, None)
    for i in range(n):
        for j in range(i+1, n):
            d = dist3(pts[i], pts[j])
```

main ⊗ 0 ⚠ 0 SantiQ (3 minutes ago) Ln 23, Col 42 Spaces: 4 UTF-8 CRLF {} Python .venv (3.13.9) Go Live

File Edit Selection View Go Run Terminal Help

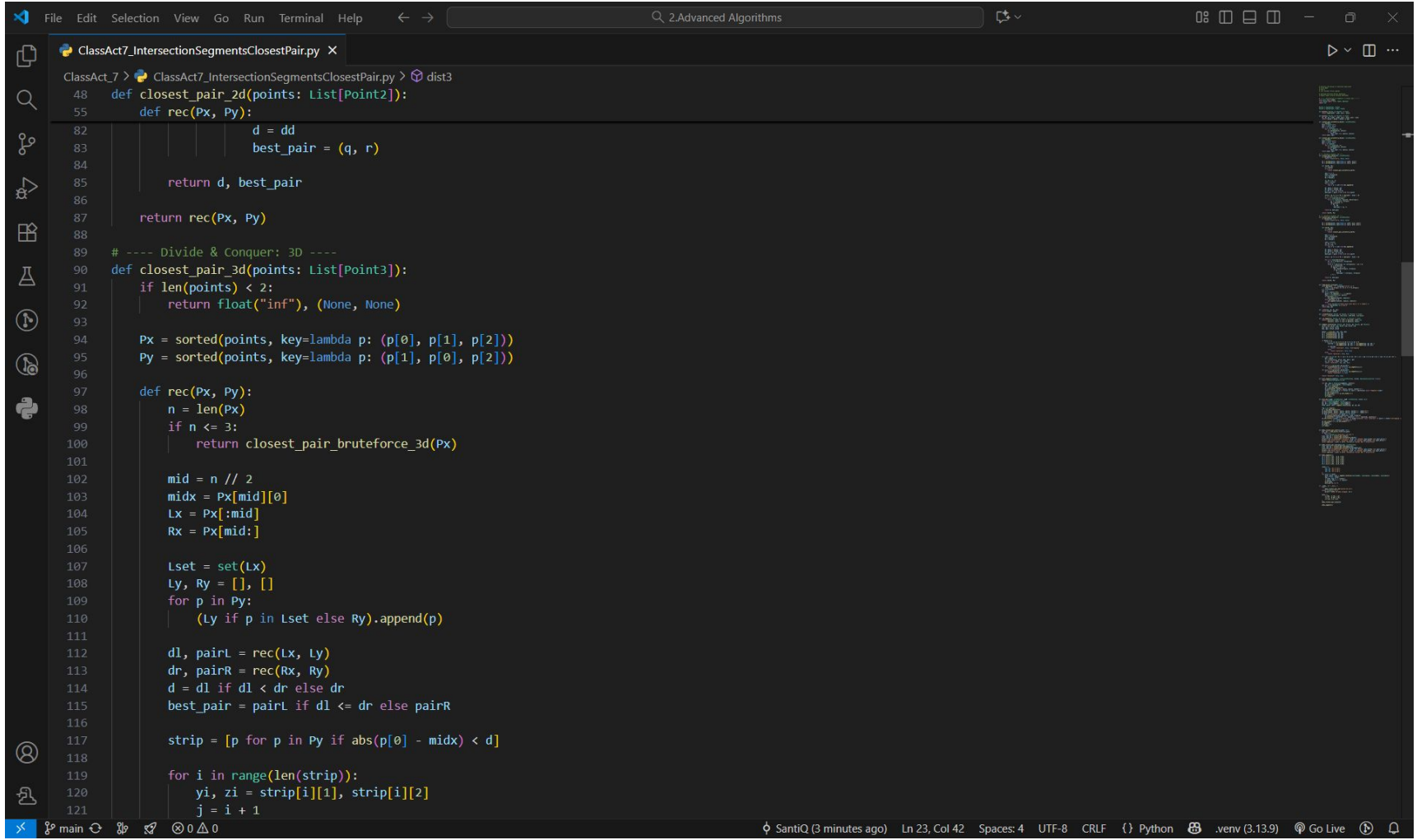
2.Advanced Algorithms

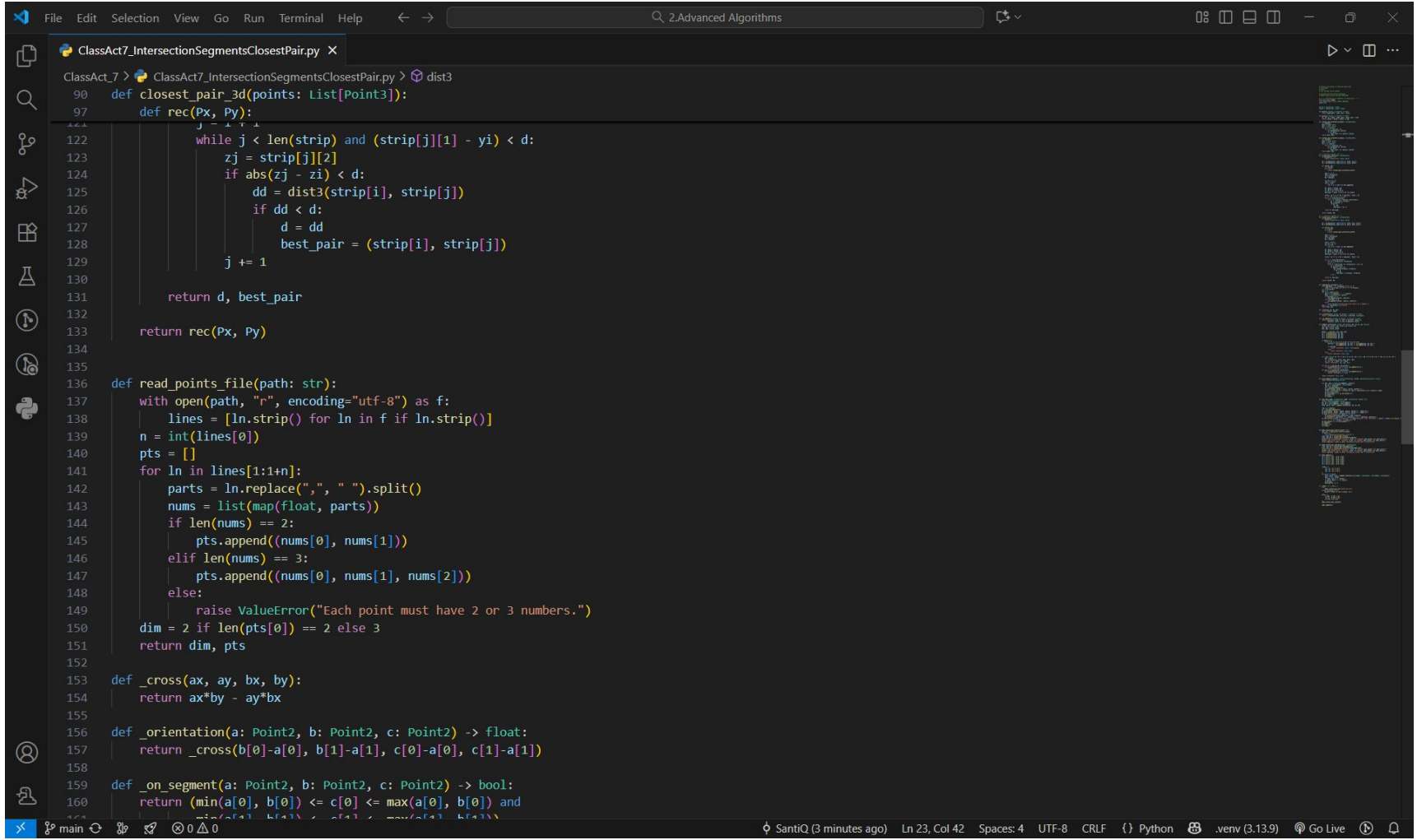
ClassAct7_IntersectionSegmentsClosestPair.py

ClassAct_7 > ClassAct7_IntersectionSegmentsClosestPair.py > dist3

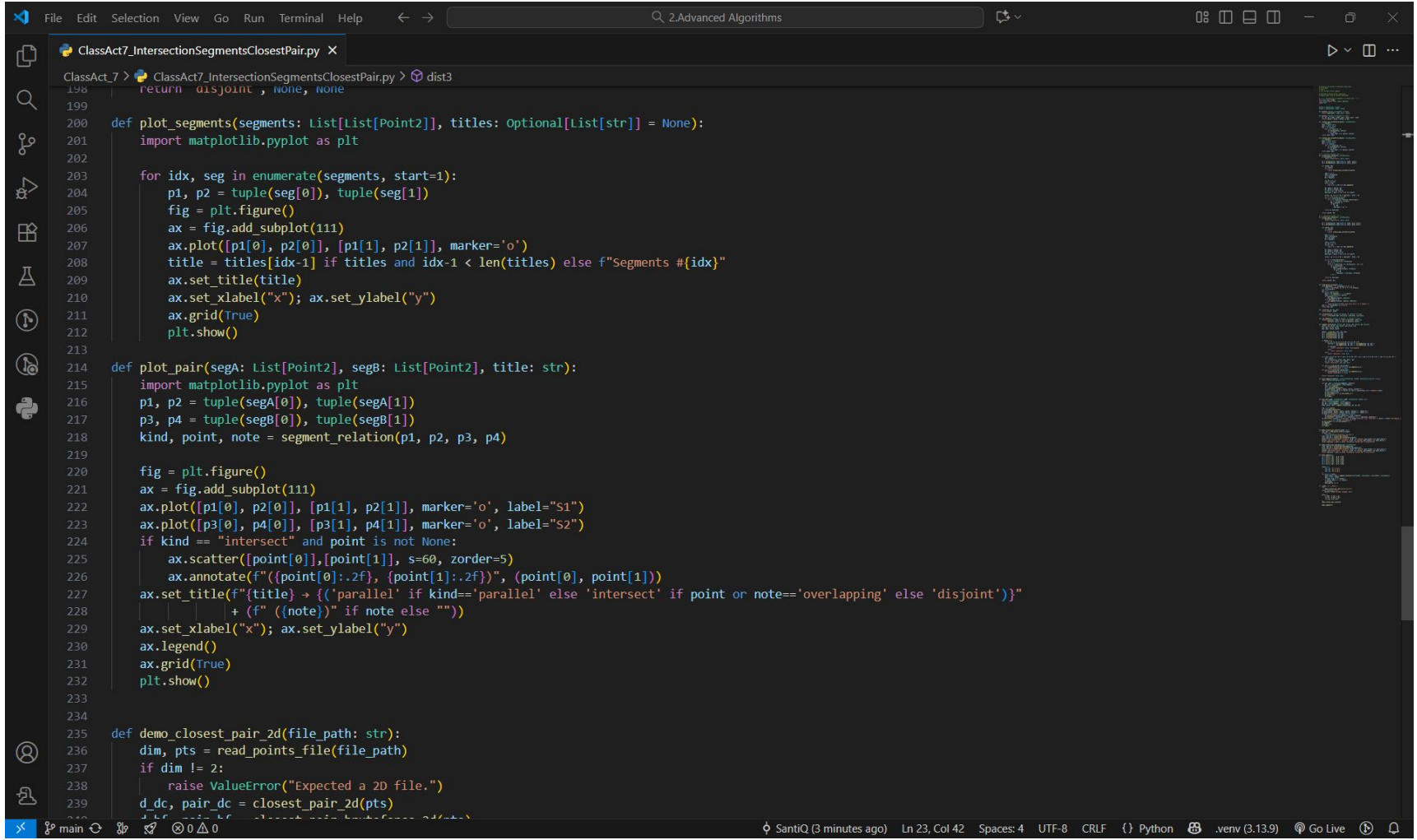
```
36  def closest_pair_bruteforce_3d(pts: List[Point3]):
42              d = dist3(pts[i], pts[j])
43              if d < best:
44                  best, pair = d, (pts[i], pts[j])
45      return best, pair
46
47  # ---- Divide & Conquer: 2D ----
48  def closest_pair_2d(points: List[Point2]):
49      if len(points) < 2:
50          return float("inf"), (None, None)
51
52      Px = sorted(points, key=lambda p: (p[0], p[1]))
53      Py = sorted(points, key=lambda p: (p[1], p[0]))
54
55      def rec(Px, Py):
56          n = len(Px)
57          if n <= 3:
58              return closest_pair_bruteforce_2d(Px)
59
60          mid = n // 2
61          midx = Px[mid][0]
62          Lx = Px[:mid]
63          Rx = Px[mid:]
64
65          Ly, Ry = [], []
66          Lset = set(Lx)
67          for p in Py:
68              (Ly if p in Lset else Ry).append(p)
69
70          dl, pairL = rec(Lx, Ly)
71          dr, pairR = rec(Rx, Ry)
72          d = dl if dl < dr else dr
73          best_pair = pairL if dl <= dr else pairR
74
75          strip = [p for p in Py if abs(p[0] - midx) < d]
76          # <= 7 next points in y-order
77          for i in range(len(strip)):
78              for j in range(i+1, min(i+8, len(strip))):
79                  q, r = strip[i], strip[j]
80                  dd = dist2(q, r)
81                  if dd < d:
```

main SantiQ (3 minutes ago) Ln 23, Col 42 Spaces: 4 UTF-8 CRLF Python .venv (3.13.9) Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
ClassAct7_IntersectionSegmentsClosestPair.py
ClassAct_7 > ClassAct7_IntersectionSegmentsClosestPair.py > dist3
48 def closest_pair_2d(points: List[Point2]):
55 def rec(Px, Py):
82 d = dd
83 best_pair = (q, r)
85 return d, best_pair
87 return rec(Px, Py)
89 # ---- Divide & Conquer: 3D ----
90 def closest_pair_3d(points: List[Point3]):
91 if len(points) < 2:
92 return float("inf"), (None, None)
94 Px = sorted(points, key=lambda p: (p[0], p[1], p[2]))
95 Py = sorted(points, key=lambda p: (p[1], p[0], p[2]))
97 def rec(Px, Py):
98 n = len(Px)
99 if n <= 3:
100 return closest_pair_bruteforce_3d(Px)
102 mid = n // 2
103 midx = Px[mid][0]
104 Lx = Px[:mid]
105 Rx = Px[mid:]
107 Lset = set(Lx)
108 Ly, Ry = [], []
109 for p in Py:
110 (Ly if p in Lset else Ry).append(p)
112 dl, pairL = rec(Lx, Ly)
113 dr, pairR = rec(Rx, Ry)
114 d = dl if dl < dr else dr
115 best_pair = pairL if dl <= dr else pairR
117 strip = [p for p in Py if abs(p[0] - midx) < d]
119 for i in range(len(strip)):
120 yi, zi = strip[i][1], strip[i][2]
121 j = i + 1
main
0 0
SantiQ (3 minutes ago)
Ln 23, Col 42
Spaces: 4
UTF-8
CRLF
Python
.venv (3.13.9)
Go Live


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
ClassAct7_IntersectionSegmentsClosestPair.py
ClassAct_7 > ClassAct7_IntersectionSegmentsClosestPair.py > dist3
def closest_pair_3d(points: List[Point3]):
def rec(Px, Py):
while j < len(strip) and (strip[j][1] - yi) < d:
zj = strip[j][2]
if abs(zj - zi) < d:
dd = dist3(strip[i], strip[j])
if dd < d:
d = dd
best_pair = (strip[i], strip[j])
j += 1
return d, best_pair
return rec(Px, Py)
def read_points_file(path: str):
with open(path, "r", encoding="utf-8") as f:
lines = [ln.strip() for ln in f if ln.strip()]
n = int(lines[0])
pts = []
for ln in lines[1:1+n]:
parts = ln.replace(",", " ").split()
nums = list(map(float, parts))
if len(nums) == 2:
pts.append((nums[0], nums[1]))
elif len(nums) == 3:
pts.append((nums[0], nums[1], nums[2]))
else:
raise ValueError("Each point must have 2 or 3 numbers.")
dim = 2 if len(pts[0]) == 2 else 3
return dim, pts
def _cross(ax, ay, bx, by):
return ax*by - ay*bx
def _orientation(a: Point2, b: Point2, c: Point2) -> float:
return _cross(b[0]-a[0], b[1]-a[1], c[0]-a[0], c[1]-a[1])
def _on_segment(a: Point2, b: Point2, c: Point2) -> bool:
return (min(a[0], b[0]) <= c[0] <= max(a[0], b[0]) and
main
SantiQ (3 minutes ago)
Ln 23, Col 42
Spaces: 4
UTF-8
CRLF
Python
.venv (3.13.9)
Go Live

```python
def _on_segment(a: Point2, b: Point2, c: Point2) -> bool:
    return (min(a[0], b[0]) <= c[0] <= max(a[0], b[0]) and
            min(a[1], b[1]) <= c[1] <= max(a[1], b[1]))

def segment_relation(p1: Point2, p2: Point2, p3: Point2, p4: Point2):
    x1,y1 = p1; x2,y2 = p2; x3,y3 = p3; x4,y4 = p4
    dx1, dy1 = x2-x1, y2-y1
    dx2, dy2 = x4-x3, y4-y3

    denom = _cross(dx1, dy1, dx2, dy2)
    o1 = _orientation(p1, p2, p3)
    o2 = _orientation(p1, p2, p4)
    o3 = _orientation(p3, p4, p1)
    o4 = _orientation(p3, p4, p2)

    if denom == 0:
        if o1 == 0 and o2 == 0 and o3 == 0 and o4 == 0:
            overlap = (_on_segment(p1, p2, p3) or _on_segment(p1, p2, p4) or
                       _on_segment(p3, p4, p1) or _on_segment(p3, p4, p2))
            if overlap:
                return "intersect", None, "overlapping"
            else:
                return "parallel", None, None
        else:
            return "parallel", None, None

    if ( (o1 > 0 and o2 < 0) or (o1 < 0 and o2 > 0) ) and ( (o3 > 0 and o4 < 0) or (o3 < 0 and o4 > 0) ):
        det = denom
        t = _cross(x3-x1, y3-y1, dx2, dy2) / det
        ix, iy = x1 + t*dx1, y1 + t*dy1
        return "intersect", (ix, iy), None

    for a,b,c in [(p1,p2,p3),(p1,p2,p4)]:
        if _orientation(a,b,c) == 0 and _on_segment(a,b,c):
            return "intersect", c, None
    for a,b,c in [(p3,p4,p1),(p3,p4,p2)]:
        if _orientation(a,b,c) == 0 and _on_segment(a,b,c):
            return "intersect", c, None

    return "disjoint", None, None

def plot_segments(segments: List[List[Point2]], titles: Optional[List[str]] = None):
```


File Edit Selection View Go Run Terminal Help
2.Advanced Algorithms
ClassAct7_IntersectionSegmentsClosestPair.py
ClassAct_7 > ClassAct7_IntersectionSegmentsClosestPair.py > dist3
return "disjoint", None, None
def plot_segments(segments: List[List[Point2]], titles: Optional[List[str]] = None):
import matplotlib.pyplot as plt
for idx, seg in enumerate(segments, start=1):
p1, p2 = tuple(seg[0]), tuple(seg[1])
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot([p1[0], p2[0]], [p1[1], p2[1]], marker='o')
title = titles[idx-1] if titles and idx-1 < len(titles) else f"Segments #{idx}"
ax.set_title(title)
ax.set_xlabel("x"); ax.set_ylabel("y")
ax.grid(True)
plt.show()
def plot_pair(segA: List[Point2], segB: List[Point2], title: str):
import matplotlib.pyplot as plt
p1, p2 = tuple(segA[0]), tuple(segA[1])
p3, p4 = tuple(segB[0]), tuple(segB[1])
kind, point, note = segment_relation(p1, p2, p3, p4)
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot([p1[0], p2[0]], [p1[1], p2[1]], marker='o', label="S1")
ax.plot([p3[0], p4[0]], [p3[1], p4[1]], marker='o', label="S2")
if kind == "intersect" and point is not None:
ax.scatter([point[0]],[point[1]], s=60, zorder=5)
ax.annotate(f"({point[0]:.2f}, {point[1]:.2f})", (point[0], point[1]))
ax.set_title(f"{title} → {('parallel' if kind=='parallel' else 'intersect' if point or note=='overlapping' else 'disjoint')}"
+ (f" ({note})" if note else ""))
ax.set_xlabel("x"); ax.set_ylabel("y")
ax.legend()
ax.grid(True)
plt.show()
def demo_closest_pair_2d(file_path: str):
dim, pts = read_points_file(file_path)
if dim != 2:
raise ValueError("Expected a 2D file.")
d_dc, pair_dc = closest_pair_2d(pts)
main
SantiQ (3 minutes ago)
Ln 23, Col 42
Spaces: 4
UTF-8
CRLF
Python
.venv (3.13.9)
Go Live

File Edit Selection View Go Run Terminal Help

2.Advanced Algorithms

ClassAct7_IntersectionSegmentsClosestPair.py

ClassAct_7 > ClassAct7_IntersectionSegmentsClosestPair.py > dist3

```python
def demo_closest_pair_2d(file_path: str):
    d_dc, pair_dc = closest_pair_2d(pts)
    d_bf, pair_bf = closest_pair_bruteforce_2d(pts)
    print(f"[2D] Divide&Conquer distance: {d_dc:.6f} between {pair_dc[0]} and {pair_dc[1]}")
    print(f"[2D] Brute force   distance: {d_bf:.6f} between {pair_bf[0]} and {pair_bf[1]}")
    assert abs(d_dc - d_bf) <= 1e-9, "Validation failed: D&C != brute-force"

def demo_closest_pair_3d(sample_pts: List[Point3]):
    d_dc, pair_dc = closest_pair_3d(sample_pts)
    d_bf, pair_bf = closest_pair_bruteforce_3d(sample_pts)
    print(f"[3D] Divide&Conquer distance: {d_dc:.6f} between {pair_dc[0]} and {pair_dc[1]}")
    print(f"[3D] Brute force   distance: {d_bf:.6f} between {pair_bf[0]} and {pair_bf[1]}")
    assert abs(d_dc - d_bf) <= 1e-9, "Validation failed: D&C != brute-force"

def demo_segments():
    S1 = [[1.0, 2.0],  [3.0, 4.0]]
    S2 = [[2.0, 2.0],  [4.0, 1.0]]
    S3 = [[1.0, 4.0],  [3.0, 6.0]]
    S4 = [[2.0, 4.0],  [4.0, 2.0]]
    S5 = [[1.0, 1.0],  [4.0, 4.0]]
    S6 = [[1.0, 8.0],  [2.0, 4.0]]

    cases = [
        (S1, S2, "S1 vs S2"),
        (S3, S4, "S3 vs S4"),
        (S5, S6, "S5 vs S6"),
    ]
    for a,b,t in cases:
        kind, point, note = segment_relation(tuple(a[0]), tuple(a[1]), tuple(b[0]), tuple(b[1]))
        msg = f"{t}: {kind}"
        if note: msg += f" ({note})"
        if point: msg += f" at {point}"
        print(msg)
        plot_pair(a, b, t)

if __name__ == "__main__":
    try:
        demo_closest_pair_2d("puntos-n8.txt")
    except Exception as e:
        print(f"[WARN] 2D demo skipped: {e}")

    pts3 = [
```

main ⊗ 0 ⚠ 0 SantiQ (3 minutes ago) Ln 23, Col 42 Spaces: 4 UTF-8 CRLF {} Python .venv (3.13.9) Go Live

File Edit Selection View Go Run Terminal Help

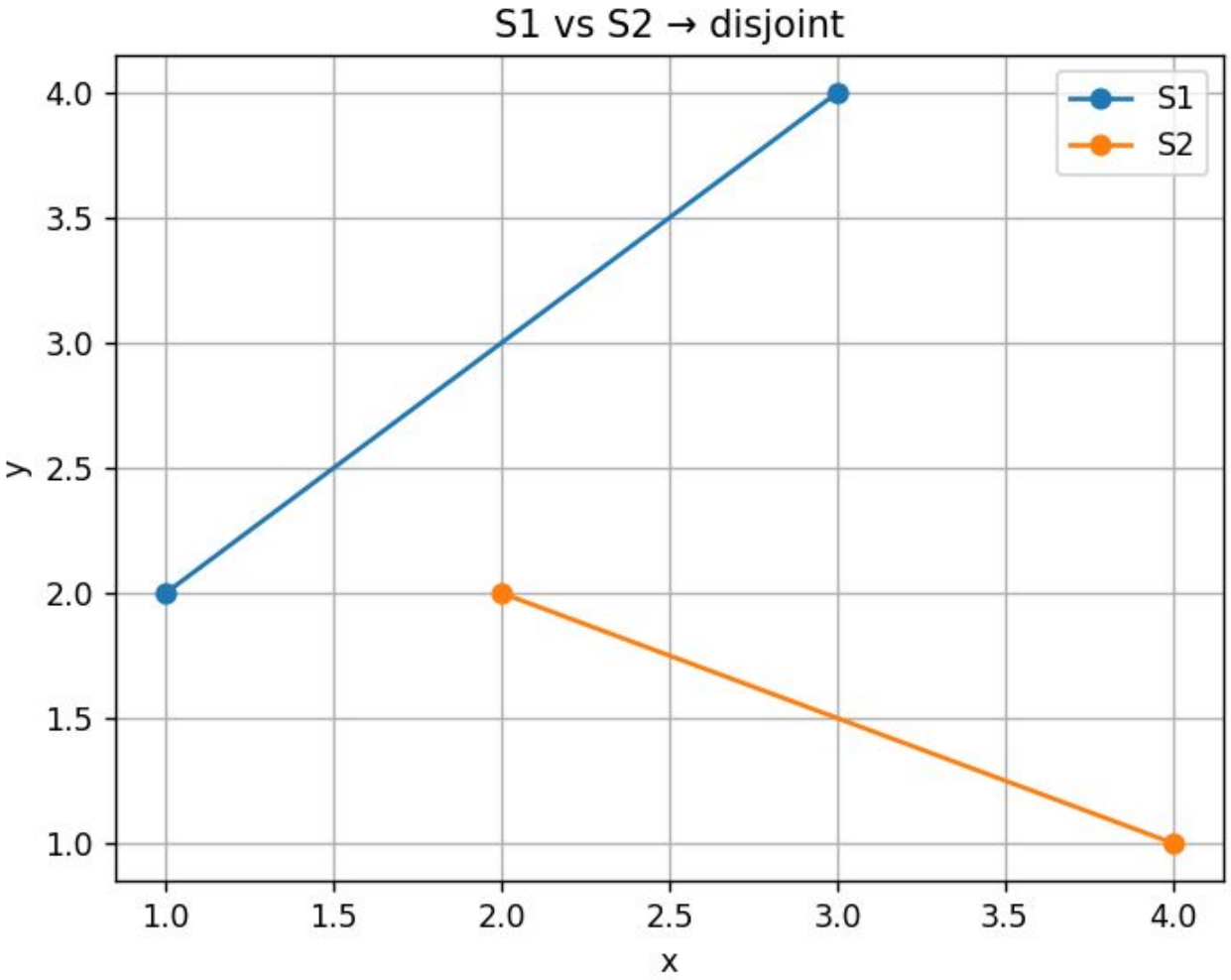
2.Advanced Algorithms

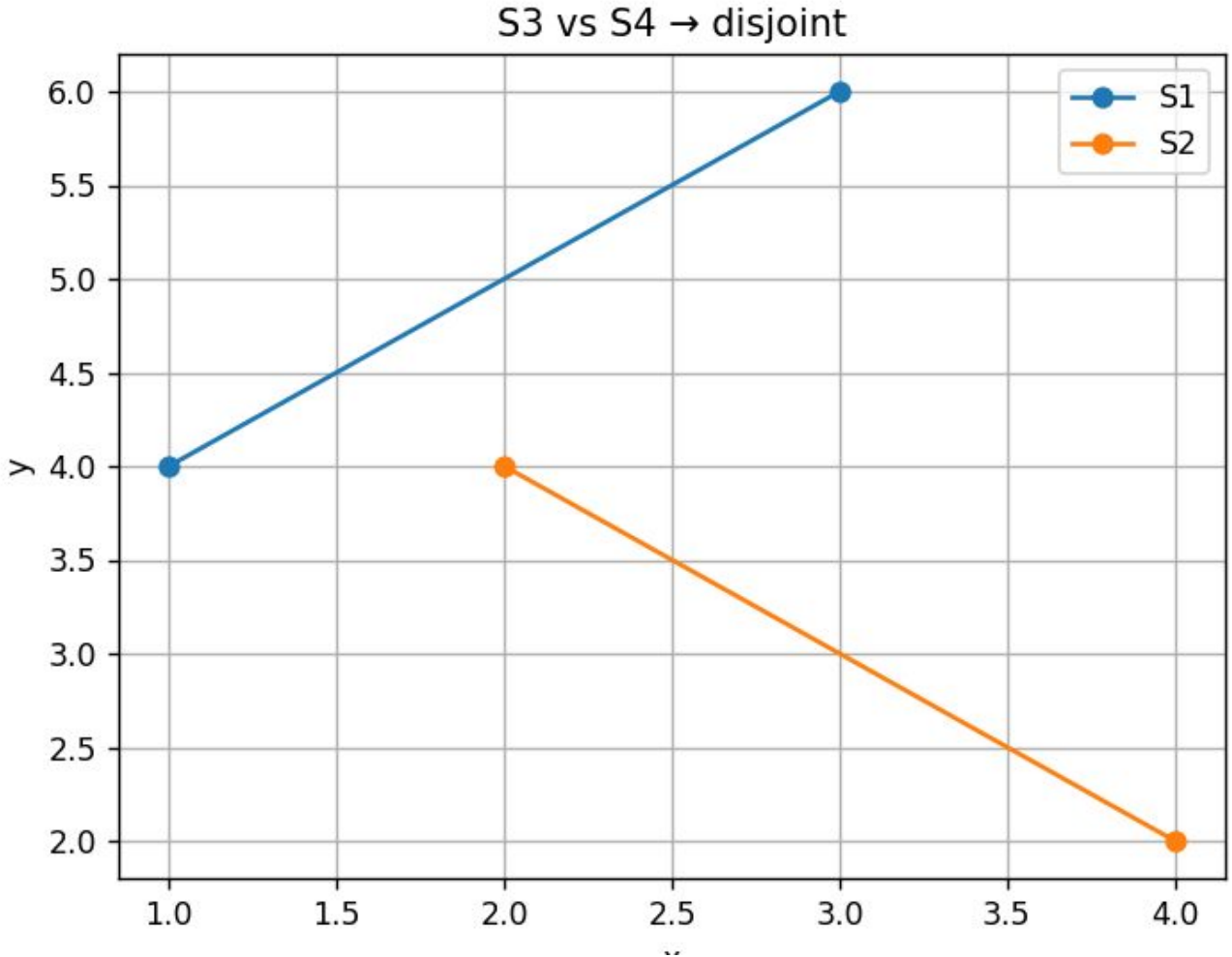
ClassAct7_IntersectionSegmentsClosestPair.py

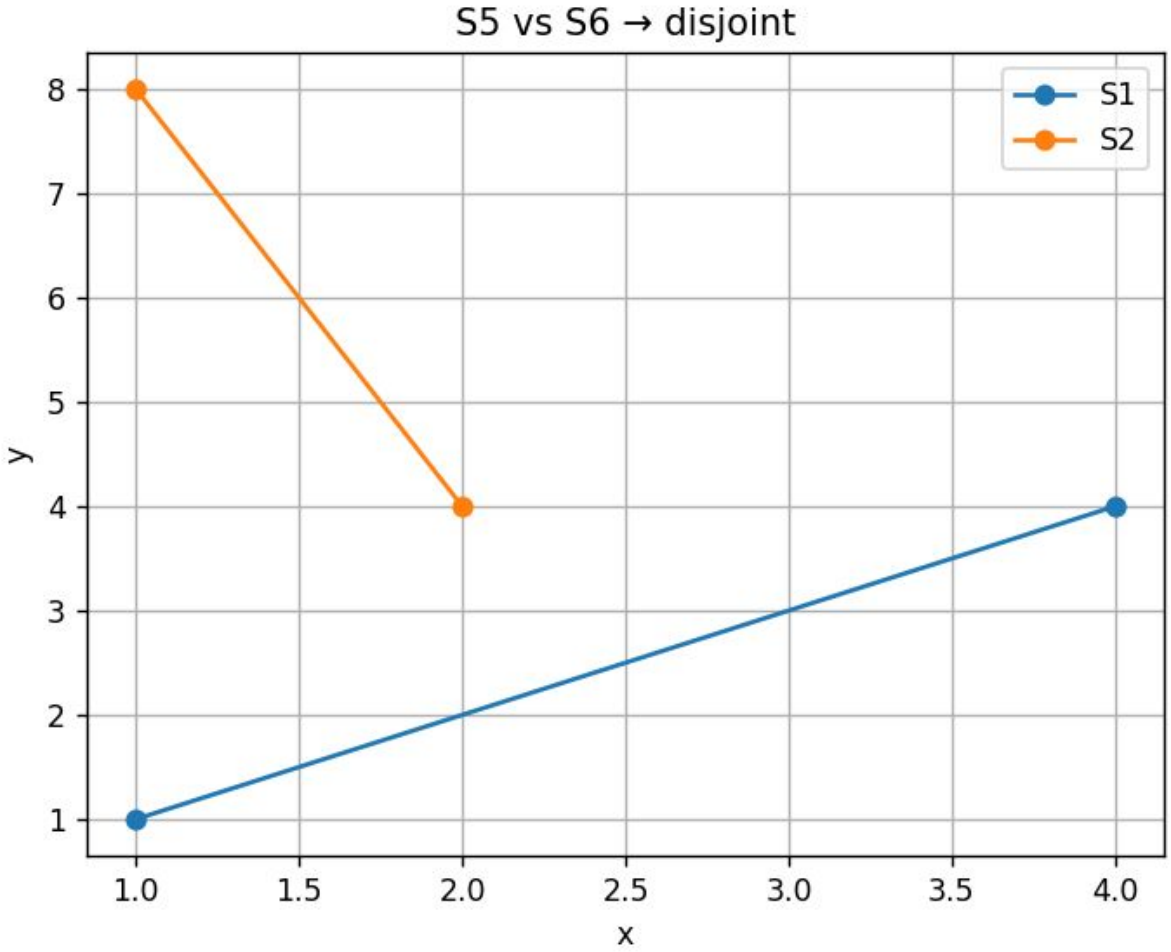
ClassAct_7 > ClassAct7_IntersectionSegmentsClosestPair.py > dist3

```python
def demo_segments():
    S5 = [[1.0, 1.0],  [4.0, 4.0]]
    S6 = [[1.0, 8.0],  [2.0, 4.0]]

    cases = [
        (S1, S2, "S1 vs S2"),
        (S3, S4, "S3 vs S4"),
        (S5, S6, "S5 vs S6"),
    ]
    for a,b,t in cases:
        kind, point, note = segment_relation(tuple(a[0]), tuple(a[1]), tuple(b[0]), tuple(b[1]))
        msg = f"{t}: {kind}"
        if note: msg += f" ({note})"
        if point: msg += f" at {point}"
        print(msg)
        plot_pair(a, b, t)

if __name__ == "__main__":
    try:
        demo_closest_pair_2d("puntos-n8.txt")
    except Exception as e:
        print(f"[WARN] 2D demo skipped: {e}")

    pts3 = [
        (2.408, -5.758, 1.0),
        (-2.77, -0.026, 5.0),
        (-7.757, 5.6, 6.0)
    ]
    demo_closest_pair_3d(pts3)

    demo_segments()
```

main 0 0 SantiQ (3 minutes ago) Ln 23, Col 42 Spaces: 4 UTF-8 CRLF Python .venv (3.13.9) Go Live


S1 vs S2 → disjoint
S1
S2
y
x
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0


S3 vs S4 → disjoint
S1
S2
y
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0


S5 vs S6 → disjoint
S1
S2
y
x
8
7
6
5
4
3
2
1
1.0
1.5
2.0
2.5
3.0
3.5
4.0

```
(.venv) PS D:\1.SQM\1.UNIVERSIDAD\5. QUINTO SEMESTRE\2.Advanced Algorithms> & "D:\1.SQM\1.UNIVERSIDAD\
ERSIDAD/5. QUINTO SEMESTRE/2.Advanced Algorithms/ClassAct_7/ClassAct7_IntersectionSegmentsClosestPair.
[WARN] 2D demo skipped: [Errno 2] No such file or directory: 'puntos-n8.txt'
[3D] Divide&Conquer distance: 7.584329 between (-7.757, 5.6, 6.0) and (-2.77, -0.026, 5.0)
[3D] Brute force   distance: 7.584329 between (-2.77, -0.026, 5.0) and (-7.757, 5.6, 6.0)
S1 vs S2: disjoint
S3 vs S4: disjoint
S5 vs S6: disjoint
```

https://colab.research.google.com/drive/1YU1SL-qp27ZECpn46oDAJCJBdIhKv3Ib?usp=sharing

REFERENCES

GeeksforGeeks. (2025, August 27). *KMP algorithm for pattern searching*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/kmp-algorithm-for-pattern-searching/

GeeksforGeeks. (2025, August 5). *Z algorithm (Linear time pattern searching Algorithm)*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/z-algorithm-linear-time-pattern-searching-algorithm/

GeeksforGeeks. (2024, April 20). *Naive algorithm for Pattern Searching*. GeeksforGeeks. https://www.geeksforgeeks.org/dsa/naive-algorithm-for-pattern-searching/