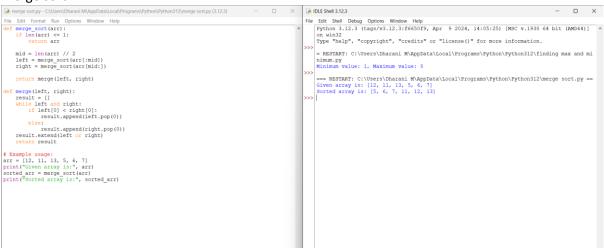
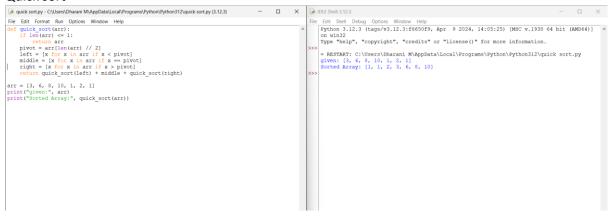
1. Finding the maximum and minimum

```
A finding max and minimum.py - C\Users\Dharani M\AppData\loca\Programs\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\Python\P
```

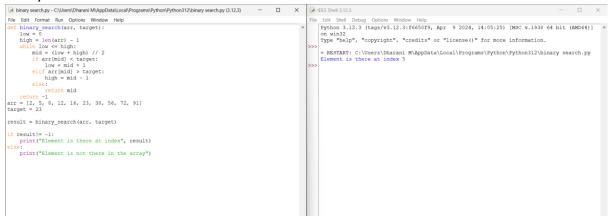
2. Merge sort



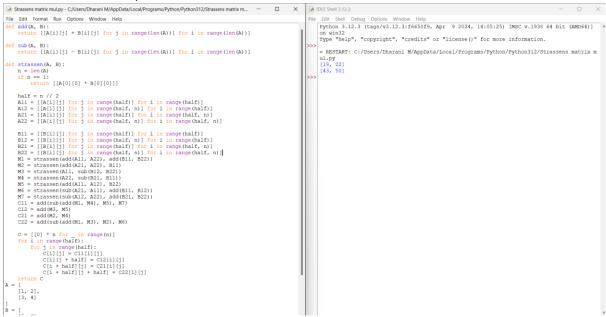
3. Quick sort



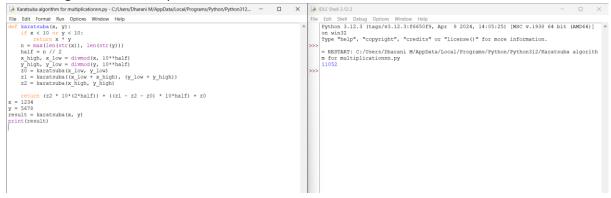
4. Binary search



5. Strassens matrix multiplication



6. ratsuba algorithm for multiplication



7. Closest pair of points using divide and conquer

🗦 *Closest pair of points using divide and conquer.py - C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Closest pair of poi File Edit Format Run Options Window Help def eucl(p, q): return ((p[0] - q[0])*2 + (p[1] - q[1])2)*0.5 def brute(points): min_dist = float('inf') best_pair = (None, None) for i in range(len(points)): for j in range(i + 1, len(points)): dist = eucl(points[i], points[j]) if dist < min dist:</pre> min dist = dist best pair = (points[i], points[j]) return best_pair[0], best_pair[1], min_dist def closesplit(points, delta): n = len(points) $mid_x = points[n // 2][0]$ $strip = [p for p in points if abs(p[0] - mid_x) < delta]$ strip.sort(key=lambda p: p[1]) best dist = delta best_pair = (None, None) for i in range(len(strip)): for j in range(i + 1, min(i + 7, len(strip))): dist = eucl(strip[i], strip[j]) if dist < best dist:</pre> best dist = dist best_pair = (strip[i], strip[j]) return best pair[0], best pair[1], best dist def recursive(points): n = len(points)if n <= 3: return brute (points) mid = n // 2Q = points[:mid] R = points[mid:] (p1, q1, dist1) = recursive(Q) (p2, q2, dist2) = recursive(R)

8. Median of medians

delta = min(dist1, dist2)

(p3, q3, dist3) = closesplit(points, delta)

9. Meet in middle technique

