**Problem 1 (10 points).** Solve each of the following recursions.

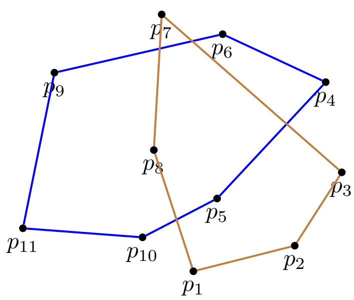
1. $T(n) = 2 \cdot T(n/2) + n \cdot \log n$
2. $T(n) = 4 \cdot T(n/2) + n \cdot (\log n)^2$
3. $T(m,n) = 4 \cdot T(m, n/2) + m \cdot n^2$

**Problem 2 (10 points).** You are given a polygon with $n$ vertices, represented as the coordinates of its $n$ vertices $((x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n))$ along the polygon in counter-clockwise order. Design a linear-time algorithm to decide whether this polygon is convex.

**Problem 3 (20 points).** You are given a sorted array $S[1 \cdots n]$ with $n$ distinct integers, i.e., $S[i] < S[i+1]$, for all $1 \le i < n$. Design a divide-and-conquer algorithm to decide whether there exists an index $k$ such that $S[k] = k$. Your algorithm should run in $O(\log n)$ time.

**Problem 4 (20 points).** Given the following two convex polygons $C_1 = (p_1, p_2, p_3, p_7, p_8)$ and $C_2 = (p_5, p_4, p_6, p_9, p_{11}, p_{10})$, compute the convex hull of $C_1 \cup C_2$ using the linear time algorithm described within the divide-and-conquer algorithm for convex hull:

1. Partition $C_2$ into two sorted list, $C_2^{UP}$ and $C_2^{LOW}$, such that the points in each list are sorted w.r.t. the anchor point $p_1$ in counter-clockwise order.
2. Give the merged list of $C_2^{UP}$ and $C_2^{LOW}$, denoted as $C_2'$, such that all points in $C_2'$ are sorted w.r.t. the anchor point $p_1$ in counter-clockwise order.
3. Give the merged list $C_2'$ and $C_1$, denoted as $C$, such that all points in $C$ are sorted w.r.t. the anchor point $p_1$ in counter-clockwise order.
4. Run the Graham-Scan-Core algorithm with $C$ as input: give the status of the stack as each point in $C$ gets processed.




**Problem 5 (10 points).** Analysis the expected running time of the following randomized algorithm for sorting. You may assume that all elelments in $A$ are distinct.

---

function combined-sort (array $A[1 \cdots n]$)
    if $n = 1$, then return $A$;
    select $k$ from $\{1, 2, \cdots, n\}$ uniformly at random;
    compute $A_L$ as the list of elements in $A$ that are smaller than $A[k]$;
    compute $A_R$ as the list of elements in $A$ that are larger than $A[k]$;
    $X_L =$ merge-sort $(A_L)$;
    $X_R =$ merge-sort $(A_R)$;
    return $(X_L, A[k], X_R)$.
end function

---

**Problem 6 (30 points).** The square of a matrix $A$ is its product with itself, i.e., $AA$.

1. Show that 5 multiplications are sufficient to compute the square of a $2 \times 2$ matrix.

2. What is wrong with the following algorithm for computing the square of an $n \times n$ matrix?
*"Use a divide-and-conquer algorithm as in Strassen's algorithm, except that instead of getting 7 subproblems of size $n/2$, we now get 5 subproblems of size $n/2$ thanks to part (1). Using the same analysis as in Strassen's algorithm, we can conclude that the algorithm runs in time $O(n^{\log_2 5})$"*

3. Show that if $n \times n$ matrices can be squared in $O(n^c)$ time for certain constant $c$, then any two $n \times n$ matrices can be multipled in $O(n^c)$.