Outline

Wrap up MergeSort complexity

Divide and Conquer

Using the Master Theorem

Notes
Recall MergeSort

MergeSort(A,b,e):
    if b == e: return
    m = (b + e) / 2
    MergeSort(A,b,m)
    MergeSort(A,m+1,e)
    # merge sorted A[b..m] and A[m+1..e] back into A[b..e]
    for i in [b,...,e]: B[i] = A[i]
    c = b
    d = m+1
    for in [b,...,e]:
        if d > e or (c <= m and B[c] < B[d]):
            A[i] = B[c]
            c = c + 1
        else: # d <= e and (c > m or B[c] >= B[d])
            A[i] = B[d]
            d = d + 1
Merge Sort

Upper bound on $T(n)$
Merge Sort

Upper bound on $T(n)$
Class of algorithms: partition problem into $b$ roughly equal subproblems, solve, and recombine:

$$T(n) = \begin{cases} 
    k & \text{if } n \leq B \\
    a_1 T([n/b]) + a_2 T([n/b]) + f(n) & \text{if } n > B
\end{cases}$$

where $B, k > 0, b > 1, a_1, a_2 \geq 0$, and $a = a_1 + a_2 > 0$. $f(n)$ is the cost of splitting and recombining.
Master Theorem

If $f$ from the previous slide has $f \in \theta(n^d)$, then

$$T(n) \in \begin{cases} 
\theta(n^d) & \text{if } a < b^d \\
\theta(n^d \log n) & \text{if } a = b^d \\
\theta(n^{\log_b a}) & \text{if } a > b^d
\end{cases}$$
Applying the Master Theorem

MergeSort
Applying the Master Theorem
RecBinSearch
Closest Point Pairs

see Wikipedia
Dividing and Conquering ClosestPointPairs
Dividing and Conquering Closest Point Pairs
Dividing and Conquering ClosestPointPairs
Dividing and Conquering Closest Point Pairs
Dividing and Conquering ClosestPointPairs
Algorithm for ClosestPointPairs

ClosestPairRec(P_x, P_y):
    if |P| \leq 3:
        find closest points by brute force
    else:
        construct Q_x, Q_y, R_x, R_y
        (q_0,q_1) = ClosestPairRec(Q_x, Q_y)
        (r_0,r_1) = ClosestPairRec(R_x, R_y)
        \( \delta = \min( d(q_0,q_1), d(r_0,r_1) ) \)
        m = average of rightmost x-coordinate in Q
        and leftmost x-coordinate in R
        construct S_x, S_y
        for each s \in S_y:
            compute distance to next 15 points in S_y
            and let (s_0,s_1) be closest pair found
Recurrence for ClosestPointPairs
Applying the Master Theorem