CS3230
Tutorial 5

1. Give an algorithm which finds the distance between closest pair of points, when all the points lie on a straight line (though not necessarily all on the x axis). Try to make the algorithm simpler than the one done in class.

   Ans:

   (A) Sort the points based on x-coordinate (as long as the points are not on a line perpendicular to x axis).

   (B) Find distances between adjacent points in the above sorted order, and report the shortest distance among these.

   In case the points are on a line perpendicular to x axis, then sort them based on y axis, and do (B) above.

2. Suppose we are sorting an array consisting of integers ≥ 0. Explain how one may modify the inputs (in linear time) so that any sorting algorithm becomes a stable sorting algorithm.

   Your modification of the numbers should be such that one can easily obtain the original numbers back from the modification.

   Ans: (A) Suppose there are n integers in the array A.

   For $i = 1$ to n: let $A[i] = A[i] * (n + 1) + i$.

   (B) Sort the array A

   (C) For $j = 1$ to n: let $A[j] = \lfloor \frac{A[j]}{n+1} \rfloor$.

3. Suppose A is a sorted array of n **distinct** integers. Give an algorithm to find if there exists an i such that $A[i] = i$ (and in case such an i exists, then return one such i).

   For example, if the array contains $-4, -2, 2, 4, 7$, then $A[4] = 4$.

   One can find such an i by just checking whether $A[j] = j$, for each j between 1 and n (both inclusive). However this is $O(n)$ algorithm.

   Can you give a better algorithm? Give time complexity (worst case) analysis of your algorithm.

   Ans: The following algorithm solves the question.

Find-id(A, i, j)

(* Initially the above algorithm is called with $i = 1$ and $j = n$; Iteratively, we know that only r such that $i \leq r \leq j$ may satisfy the property that $A[r] = r$. *).

1. If $i = j$, and $A[i] = i$, then return i.
2. If ($i = j$ and $A[i] \neq i$) or $i > j$), then return NONE.
3. Otherwise, let $m = \lfloor \frac{j+i}{2} \rfloor$.
   If $A[m] = m$, then return m.
   Else If $A[m] > m$, then return Find-id$(A, i, m-1)$.
   Else if $A[m] < m$, then return Find-id$(A, m+1, j)$.

Complexity:

$T(n) \leq T(n/2) + C$.

Solving the recurrence gives $T(n) = O(\log n)$.

4. Suppose A is adjacency matrix of a simple graph. Show that $A^m[i, j]$ gives number of paths from i to j of length exactly m

   (here A^0 is the identity matrix (which has 1 in the diagonals and 0 everywhere else), $A^1 = A$, and $A^{n+1} = A^n * A$).

   Ans: By induction.

   Base case: $A[i, j]$ gives the number of paths of length 1 from i to j.

   Induction case: Suppose $A^m[i, j]$ gives number of paths from i to j of length m.

   Consider $A^{m+1}[i, j] = \sum_{k=1}^{n} A^m[i, k] * A[k, j]$.

   Now, consider any path from i to j. For each edge (k, j) in the graph, consider the number of paths from i to j, of length $m + 1$, with the last edge being (k, j). The number of such paths is same as the number of paths from i to k of length $m - 1$.

   It is thus easy to verify that $A^{m+1}[i, j]$ gives the number of paths from i to j of length $m + 1$.

5. Consider the following coin denominations. For each part, say whether the greedy algorithm done in class gives an optimal answer. If so, give an argument to justify your answer. If not, then give a counterexample.

   (a) 1, 5, 7

   (b) 1, 5, 15

   (c) 1, 4, 9

   (d) 1, 3, 5

   Ans:

   (a) No. Counterexample: 10

   (b) Yes. Greedy algorithm is optimal for every set of denominations, $1 = d_1 < d_2 < d_3 \ldots < d_n$, where d_{i+1} is a multiple of d_i. This is so because the optimal solution can then have at most $(d_{i+1}/d_i) - 1$ number of coins of denomination d_i, which in turn implies

that the value of all the coins of denomination $< d_i$ add up to a value $< d_i$. This in turn implies that the greedy and optimal solutions have exactly the same coins.

(c) No. Counterexample: 12

(d) Yes. In any optimal solution, usage of two 3s can be replaced by one 1 and one 5. Furthermore, usage of one 3 and two 1s can be replaced by usage of one 5. Thus, in any optimal solution, the total value of coins of denominations 3 and 1 is at most 4. The same holds for greedy algorithm also. It follows that the greedy and optimal algorithm use same number of 5s. Also, easy case analysis for total value of ≤ 4, shows that greedy and optimal algorithms use same number of coins for any value ≤ 4.