CS345 : Algorithms II
Semester I, 2017-18, CSE, IIT Kanpur

Assignment 1 Solution

September 2, 2017

1 Hard Version

1.1 Non-dominated Points

1. **Output Sensitive Algorithm** (*marks=20*)

   - The divide and conquer algorithm studied in class is modified slightly such that we recurse on only those points in the left sub-space that are non-dominated by any points in the right sub-space.
   - Observe that the worst case time complexity of the algorithm occurs when the recursion tree is complete binary tree i.e. the height of the recursion tree is $\log h$.

2. **Extension to 3 dimensions** (*marks=10,10*)

   (a)
   - Note that the set of non-dominated points in 2D plane when sorted according to increasing x-coordinate they get sorted in decreasing y-coordinate.
   - Maintain a height balanced BST sorted on x(or y) coordinate of the current non-dominated points. By the above property if we look at the tree from the right to left we also get a tree sorted in y coordinate.
   - Use the BST to check if a new point being considered is non dominated or not. If non-dominanted check if it dominates any points in current set of non-dominated points and delete them. This can be done efficiently in $O(\log i)$ time by insert/delete/successor/predecessor operations of BST.
   - Observe that multiple points may get deleted from BST in a single iteration but as total number of deletions cannot exceed the total number of points, the total time complexity of deletions is $O(i \log i)$.

   (b)
   - We consider the given points according to decreasing z-coordinate. This ensures that points do not dominate points considered before them. Also, now for dominated points we only need to check for the x and y coordinates.
   - To check efficiently if the x-y projection of point is non-dominated we use the same algorithm as part (a). Also, we keep a separate list to store all non-dominated points in 3D.

1.2 A computational problem of an experimental physicist

(*marks=20*)

- Construct polynomials P and Q such that the required force on a single particle can be expressed as coefficients of polynomial $P \times Q$.
- For example consider $P(x) = \frac{1}{(n-1)^2} + \frac{1}{(n-2)^2}x + ... + \frac{1}{(1)^2}x^{(n-2)} + 0 * x^{(n-1)} + \frac{-1}{(1)^2}x^n + ... + \frac{-1}{(n-1)^2}x^{(2n-2)}$ and $Q(x) = q_n x^{n-1} + q_{n-1}x^n + ... + q_1 x^{2n-2}$. And note that the force on the jth partice is given by $C * q_j *$ coefficient of x^{3n-2-j} in $P(x) \times Q(x)$.
- Use the algorithm discussed in class to multiply polynomials efficiently.

2 Easy Version

1. **Non-dominated points in online fashion** (*marks=15*)

   - Refer to the explanation of 1.1 second part in hard version.

2. **Application of the algorithm for multiplication of polynomials** (*marks=15*)

   - Construct the polynomials P_A and P_B for the set A and B such that the coefficient of monomial x^i is 1 if element i is present in the set and 0 otherwise.
   - Now observe that the coefficient of x^i in $P_A \times P_B$ gives the number of ways we can represent i as the sum of an element from A and an element from B.
   - Use the algorithm discussed in class to multiply polynomials efficiently.

3. **Augmented binary search tree** (*marks=15*)

   - Use the balanced binary search tree augmented at node x with min(x) : minimum element of subtree rooted at x, max(x) : maximum element of subtree rooted at x and min-gap(x) : min-gap of subtree rooted at x.
   - Now note that all these augmentation properties can be maintained locally, that is by just looking at the left child, right child and the node we can figure out the augmentation properties at the current node. For example min-gap(x)=min(x.left, x.right, x-x.left.max, x.right.min -x) and similarly for others.
   - Convince yourself that because of locality of augmentation we can perform the insert and delete operations in $O(\log n)$ as we would only have to update $O(\log n)$ nodes in bottom up fashion. Min-gap query will be done trivially by reporting min-gap of root node.