Practice 4 Nesting Boxes

Problem 1

The nesting relation is transitive.

Proof: Assume that there exists permutations: π_1 such that box A nests within box B, π_2 such that box B nests within box C ($A,\ B,\ C$ are three d-dimensional boxes), we have $A_{\pi_1(1)} < B_1,\ A_{\pi_1(2)} < B_2,\ \cdots,\ A_{\pi_1(d)} < B_d$, $B_{\pi_2(1)} < C_1,\ B_{\pi_2(2)} < C_2,\ \cdots,\ B_{\pi_2(d)} < C_d$.

Therefore, there exists a permutation $\pi_3 = \pi_1 \circ \pi_2$ such that $A_{\pi_3(1)} < C_1,\ A_{\pi_3(2)} < C_2,\ \cdots,\ A_{\pi_3(d)} < C_d$.

Thus, A nests within C, which proves that the nesting relation is transitive.

Problem 2

Algorithm Steps (Pseudocode)

return value: `1` when A nests within B, `-1` when B nests within A, `0` when there is no nesting relation.

```
1  isNest(d, A, B):
2      sort(A)
3      sort(B)
4      if A[0] < B[0]:
5          then for i ← 1 to d - 1:
6              if A[i] >= B[i]:
7                  then return 0
8              return 1
9          else for i ← 0 to d - 1:
10             if A[i] <= B[i]:
11                 then return 0
12             return -1
```

Time Complexity

line 2: $O(d\ \log d)$ (Merge Sort or Quick Sort)

line 3: $O(d\ \log d)$ (Same as above)

line 4 - 12: $O(d)$

Therefore, the time complexity of this algorithm is $O(d\ \log d)$.

Correctness Proof

Sufficiency (After sorting $A_i' < B_i'$ for every $i \rightarrow A$ nests within B)

According to the definition of the nesting relation, after sorting, for every i, if $A_i' < B_i'$, A nests within B.

Necessity (A nests within $B \rightarrow$ After sorting $A_i' < B_i'$ for every i)

Basis step: When $d = 1$, since A nests within B, there is only one permutation and $A_1 < B_1$.

Inductive step: Assume that for $d = k \geq 2$, if A nests within B, then after sorting, $A_i' < B_i'$ for every i.

When $d = k + 1$, for convenience, regard the permutation that enables A to be nested within B as the original permutation. Therefore, $A_i < B_i$ for every i.

Pick out A_1 and B_1, and according to the inductive hypothesis, after sorting, $A'_i < B'_i$ for every $i > 1$.

We insert A_1 and B_1 back into the sorted permutation, in ascending order. Since $A_1 < B_1$, the position where A_1 is inserted is same as or ahead of that of B_1.

If same: The insertion doesn't affect the relationship between other A'_i and B'_i, thus the conclusion holds.

If ahead: After insertion, all A'_i after A_1 have been shifted one position to the right. Since $A_1 \leq A'_i < B'_i \leq B'_{i+1} \leq B_1$ for i within the two insertion positions, the conclusion still holds.

Problem 3

Algorithm Steps (Pseudocode)

```
1  longestNestingSequence(n, d, boxes):
2      /* nest: n*n 2-d arraylist, represent the nesting relation between box i and j
3         depth: nesting depth of box i, with initial values 0
4         prev: array with length n, store prev box of box i, with initial values -1 */
5      for i ← 0 to n - 1:
6          sort(boxes[i])
7  
8      for i ← 0 to n - 1:
9          for j ← i + 1 to n - 1:
10             if boxes[i][0] < boxes[j][0]:
11                 then for k ← 1 to d - 1:
12                     if boxes[i][k] >= boxes[j][k]:
13                         then break
14                     nest[i].add(j)
15                 else for k ← 0 to d - 1:
16                     if boxes[i][k] <= boxes[j][k]:
17                         then break
18                     nest[j].add(i)
19 
20     t = topological order of boxes
21     for i ← 0 to t.length - 1:
22         for j ← 0 to nest[j].size - 1:
23             if depth[j] < depth[i] + 1:
24                 depth[j] = depth[i] + 1
25                 prev[j] = i
26 
27     cur = index of maximum depth
28     while cur != -1:
29         print boxes[cur]
30         cur = prev[cur]
```

Time Complexity

line 5 - 6: $O(nd\ \log d)$

line 8 - 18: $O(n^2d)$

line 20 - 25: $O(n^2)$

line 27 - 30: $O(n)$

Therefore, the time complexity of this algorithm is $O(n^2d)$.