CS208 Lab4 Practice-Nesting Boxes

12312110 李轩然

DDL: Apr.13

Description

A d-dimensional box with dimensions $(x_1, x_2, \ldots, x_d)$ nests within another box with dimensions $(y_1, y_2, \ldots, y_d)$. If there exists a permutation π on $\{1, 2, \ldots, d\}$ such that $x_{\pi(1)} < y_1$, $x_{\pi(2)} < y_2$, ..., $x_{\pi(d)} < y_d$:

1. Is the nesting relation transitive? Prove it.
2. Design an efficient method to determine whether one d-dimensional box nests inside another. Describe the algorithm's steps, state its time complexity, and prove its correctness.
3. Suppose that you are given a set of n d-dimensional boxes $\{B_1, B_2, \ldots, B_n\}$. Give an efficient algorithm to find the longest sequence $\{B_{i1}, B_{i2}, \ldots, B_{ik}\}$ of boxes such that B_{ij} nests within B_{ij+1} for $j = 1, 2, \ldots, k-1$. Express the running time of your algorithm in terms of n and d.

Requirement

This week's practice does not require writing code. Instead, you need to describe the algorithmic steps (or provide pseudocode) and clearly explain the proof process.

Expected Output:

- A structured algorithm description (or pseudocode).
- A formal correctness proof (e.g., by induction, contradiction, or logical reasoning).

Analysis

1. The nesting relation is transitive

Proof: Suppose we have three boxes named A, B and C with dimensions:

$$A = (a_1, a_2, \ldots, a_d)$$

$$B = (b_1, b_2, \ldots, b_d)$$

$$C = (c_1, c_2, \ldots, c_d)$$

Assume A nests within B, B nests within C, thus we need to prove that A nests within C.

By definition, there exists a bijection $\alpha : \{1, 2, \ldots, d\} \to \{1, 2, \ldots, d\}$ such that $b_{\alpha(1)} < c_1$, $c_{\alpha(2)} < y_2$, ..., $b_{\alpha(d)} < c_d$; there also exists a bijection $\beta : \{1, 2, \ldots, d\} \to \{1, 2, \ldots, d\}$ such that $a_{\beta(1)} < b_1$, $a_{\beta(2)} < b_2$, ..., $a_{\beta(d)} < b_d$.

According to discrete mathematics, $\beta \circ \alpha$ is also bijective from $\{1, 2, \ldots, d\}$ to $\{1, 2, \ldots, d\}$, and:

$$a_{\beta\circ\alpha(1)} < b_{\alpha(1)} < c_1, a_{\beta\circ\alpha(2)} < b_{\alpha(2)} < c_2, \ldots, a_{\beta\circ\alpha(d)} < b_{\alpha(d)} < c_d$$

which means A nests within C by definiton. Then we are done.

2. Sorting

Suppose we need to determin that X nests within Y with dimensions:

$$X = (x_1, x_2, \ldots, x_d)$$

$$Y = (y_1, y_2, \ldots, y_d)$$

We just sort the dimensions of X and Y in ascending order:

$$X = (x'_1, x'_2, \ldots, x'_d), x'_1 \leq x'_2 \leq \ldots \leq x'_d$$

$$Y = (y'_1, y'_2, \ldots, y'_d), y'_1 \leq y'_2 \leq \ldots \leq y'_d$$

Then X nests within Y iff:

$$x'_1 < y'_1, x'_2 < y'_2, \ldots, x'_d < y'_d$$

The total time complexity will be $O(dlogd)$ because of sorting.

Correctness proof: Assume X nests within Y, but there exists $i \in \{1, 2, \ldots, d\}$, such that $x'_i \geq y'_i$ after sorting. Because $y'_1 \leq y'_2 \leq \ldots \leq y'_d$, then there are $n - i + 1$ elements in X that not smaller than y'_i. However, there are only $n - i$ elements in Y that greater than y'_i, thus there must exist at least one element in X, let's say $x'_j (i \le j \le d)$, matches y'_i or smaller element in Y, let's say $y'_k (1 \le k \le i)$, then $x'_j \ge y'_{k'}$, which means X does not nest within Y and makes a contridiction. So the algorithm is correct.

3. Greedy

Still sort the dimensions of each $B_i (1 \le i \le n)$ in ascending order:

$$B_i = (b'_{i1}, b'_{i2}, \ldots, b'_{id}), b'_{i1} \le b'_{i2} \le \ldots \le b'_{id}$$

then sort $B_1, B_2, \ldots, B_n$ by sorting $b'_{11}, b'_{21}, \ldots, b'_{n1}$ (if equals, then compare b'_{i2} ......) in ascending order:

$$B_1, B_2, \ldots, B_n (b'_{11} \le b'_{21} \le \ldots \le b'_{n1})$$

In this case, we don't know whether the largest box, which is B_n, can be chosen, so we need to list every B_i that probably be the largest. Define $f_i (1 \le i \le n)$ that means when the largest box is B_i, the number of boxes we can choose at most. Then the answer will be $\max\{f_1, f_2, \ldots, f_n\}$ but not f_n.

Clearly, $f_1 = 1$, and for $2 \le i \le n$, we need to find a box that can be nested within B_i, let's say B_j, and its f_j **is the largest**. Then add B_i to this sequence, thus $f_i = f_j + 1$. The pseudocode is:

```
f[1] = 1
for i : 2 to n
    temp = 0
    for j : i-1 downto 1
    if (B[j] nests within B[i])
        temp = max(temp, f[j])
    f[i] = temp + 1

return max{f[i]}
```

Running time: The double for-loop will cost $O(n^2)$ time, and every judgement (B_j nests within B_i) will do comparation for d times, and sorting will cost $O(ndlogn + ndlogd)$ time, thus the total time complexity will be $O(n^2 d)$.

Correctness proof: For every B_i, we guarantee that f_i is the maximum of the number of boxes we can choose when B_i is the largeset, and $f_1, f_2, \ldots, f_n$ lists all the possibilities of $\{B_n\}$ sequence that might be the longest, thus the answer must be the longest length.