Theory Assignment-5: ADA Winter-2024

Shobhit Raj (2022482) Vashu (2022606)

1 Pre-Processing

Input: Given n, the no. of boxes and their dimensions in form d1 X d2 X d3.

We would sort the dimensions of the boxes in increasing order, and just compare the smallest dimension of each box to smallest dimension of other boxes, the second smallest with the second smallest, and so on i.e. comparing the sorted dimensions pairwise between each pair of boxes. If the sorted dimensions of one box are strictly smaller than the respective dimensions of another box, then the first box can be nested inside the second box. This comparison can be done in O(1)-time.

Reason for doing this - Sorting the dimensions takes care of rotation. Once the dimensions are sorted, we don't need to consider all possible rotations separately, 6 permutations possible. By sorting and then comparing pairwise, we only need to check each box against every other box once, reducing the number of comparisons needed and making the algorithm more efficient.

Time - Sorting the dimensions of 1 box will take 3*log(3)-time, which is O(1)-time and this is done for n boxes, so total time for this pre processing will be O(n). This pre-processing is only done to make the comparison easier for checking if one box can be nested inside other or not.

2 Flow Network Construction

Assumption: Only 1 box can be nested inside other box.

This problem is an instance of Maximum Bipartite Matching problem which can be reduced to an instance of Maximum flow in a graph/network problem. So first, we make the Bipartite Graph with 2 independent groups A and B, each having n vertices representing the n boxes.

Steps of construction of the network flow/graph:

1. **Partition Boxes into Groups:**
   Divide the set of boxes into two independent groups, Group A and Group B as mentioned above. Let a_i and b_i represent all the n boxes in Group A and Group B respectively.
2. **Add Source and Sink:**
   Introduce a source node s and a sink node t to the graph. The source s will represent the starting point of the flow, and the sink t will represent the endpoint or the target.
3. **Create Edges from Source to Group A:**
   For each box a_i in Group A, add an edge from the source s to a_i with a capacity of 1, i.e., (s, a_i) with capacity 1, which signifies that each box in Group A represents one unique box.
4. **Create Edges from Group B to Sink:**
   For each box b_i in Group B, add an edge from b_i to the sink t with a capacity of 1, i.e., (b_i, t) with capacity 1, which signifies that each box in Group B represents one unique box.

5. **Create Edges between Group A and Group B:**
   For each combination of a_i in Group A and b_j in Group B (where $i \neq j$ to ensure a box is not nested inside itself), if b_j can be nested inside a_i, add an edge from a_i to b_j with a capacity of 1, i.e., (a_i, b_j) with capacity 1, this edge represents that a_i can contain box b_j , and the capacity of 1 ensures that each box can have at most one box nested inside it.

Interpretation of Maximum Flow:
The maximum flow of this graph represents the maximum number of boxes that can be nested inside other boxes (i.e., the maximum frequency of nesting a smaller box within a larger one), considering the constraint that each box can only contain one other box.

Computing Maximum Flow:
The maximum flow of the network can be calculated using the Ford-Fulkerson algorithm, which will determine the maximum number of boxes that can be nested inside each other.

Visibility of Boxes:
A box is considered visible if it is not nested inside another box. Given that there are n total boxes, the minimum number of visible boxes can be calculated as:

Minimum no. of visible boxes/boxes that cannot be nested = $n-$Maximum no. of boxes that can be nested

$$\text{Minimum number of visible boxes} = n - \text{Max Flow}$$

Identifying Visible Boxes:
The edges used in the maximum network flow form a maximum matching in the bipartite graph, indicating the optimal nesting configuration between Group A and Group B. Each edge in this maximum matching represents a box from Group B being nested inside a box from Group A.
To identify the visible boxes, we examine the boxes in Group B that do not have a flow of 1 from their corresponding vertex to the sink node t, i.e. are not part of this maximum matching. These boxes are not nested inside any box from Group A and are therefore considered visible.

3 Justification of reduction to Network Flow/Proof of Correctness

Any valid arrangement of the boxes translates to a matching in the graph. An edge (a_i, b_j) is part of this matching (i.e. have a flow of 1) if and only if box i is directly nested inside box j in the possible arrangement.

Claim: The maximum flow of this graph represents the maximum number of boxes that can be nested inside other boxes i.e. maximum matching in the bipartite graph indicates the optimal nesting configuration between Group A and Group B.

Proof:

Forward Direction: Optimal Flow $\Rightarrow$ Optimal Solution
Suppose the graph has a maximum flow or matching of value F. Then:

- $f^{\text{out}}(s) = F$ and $f^{\text{in}}(t) = F$
- Each outgoing edge from s to a_i has a capacity of 1, so F vertices in Group A have $f(s \to a_i) = 1$
- For those vertices, $f^{\text{in}}(a_i) = 1$, implying $f^{\text{out}}(a_i) = 1$
- Each incoming edge to t from b_j has a capacity of 1, so F vertices in Group B have $f(b_k \to t) = 1$
- For those vertices, $f^{\text{out}}(b_j) = 1$, implying $f^{\text{in}}(b_j) = 1$
- $f(a_i \to b_j)$ will be 1 for these F vertices/boxes in Group A & Group B.

Thus, the F vertices/boxes in Group B that have matching from vertices in Group A are nested inside the F vertices/boxes in Group A and is the maximum possible number of boxes that can be nested inside other.

Given a maximum matching M with a flow value F, there are F edges between Group A and Group B due to the capacity constraint of 1. We can determine the box placements by pairing each b_j with a_i for each edge $(a_i \to b_j)$ in M.

Each of the F vertices a_i in Group A and b_j in Group B is matched exactly once, adhering to the capacity constraint. This means that each box is nested inside another box only once, satisfying the problem's constraints.

This arrangement maximizes the number of nested boxes. If there were a better arrangement to place the boxes, it could be transformed (by converting the pair of nested boxes to edges) into a larger matching, leading to a contradiction. Thus, the current arrangement is optimal.

Backward Direction : Optimal Solution $\Rightarrow$ Optimal Flow
Suppose the maximum number of boxes that can be nested is F. Then, there exist F edges with flow 1 between Group A and Group B due to the constraint that only one box can be nested inside another and the capacity constraint of edges is 1.

Each of the F vertices/boxes in Group B can be nested inside the F vertices/boxes in Group A. Therefore, these F edges from Group A will have $f(a_i \to b_j) = 1$, implying $f^{\text{out}}(a_i) = 1$, which in turn implies $f^{\text{in}}(a_i) = 1$ for these F vertices in Group A. Hence, these F vertices in Group A will have $f(s \to a_i) = 1$, leading to $f^{\text{out}}(s) = F$, which is the maximum flow. Thus, the maximum flow of the network is F.

Given the maximum number of boxes that can be nested, denoted as F, and their containment relationships, we can create a maximum matching M as follows: when a box a_i contains another box b_j, we include the edge $(a_i \to b_j)$ in M, with a_i from Group A and b_j from Group B. This matching is valid because each box can house only one other box.

Consequently, F represents the total number of edges in this matching, and it's maximized through this approach. If there were a larger matching, it would suggest a more efficient arrangement of the boxes, which contradicts our assumption of F being the maximum possible number. Therefore, M indeed represents a maximum matching, with a flow value of F.

Thus, the maximum number of nested boxes, represented by F, is equal to the size of the maximum matching in the graph. The minimum count of visible boxes, which are not nested inside any other box, can be determined as the total number of boxes minus the maximum number of nested boxes, i.e., $n - F$.

4 Run Time Analysis

As stated in the pre-processing section, sorting the dimensions of the boxes takes $O(n)$-time.

The construction of the flow network involves creating the bipartite graph with two groups of vertices, A and B, and connecting them with edges based on the nesting relationships between the boxes. This can be done by running a double for loop over all the boxes, and comparing the sorted dimensions pairwise between each pair of boxes. If the sorted dimensions of one box are strictly smaller than the respective dimensions of another box, then the first box can be nested inside the second box, and we add an edge between A and B. This comparison & adding of edge can be done in $O(1)$-time, leading to a time complexity of $O(n^2)$ for running the double for loops, where n is the total number of boxes.

Now, adding the edges between Group A & Group B, in the worst case, any first arbitrary box can be nested inside every other $(n-1)$ boxes, the second box can be nested inside every other $(n-2)$ boxes, and so on. So, the number of edges between A & B can be $(n-1)+(n-2)+\ldots = O(n^2)$. Additionally, there are $2n$ edges from the source & to the sink. So, the total number of edges is at most $O(n^2)$. The total number of vertices is $2n+2$, which is $O(n)$.

Hence, the construction of the flow network takes $O(n^2)$-time.

The flow value is $f^{\text{out}}(s)$, and the maximum flow value will be obtained when every edge out of s has a flow = capacity, i.e., 1. So, the total n edges out of source to Group A making the maximum flow value at most $n-1$ or $O(n)$. The Ford-Fulkerson's algorithm runs in time $O(\text{value(flow)} \times (|V|+|E|))$-time. Here, $(|V|+|E|)$ is $O(n)+O(n^2)$, which is $O(n^2)$, and value(flow) is $O(n)$, making the running time of Ford-Fulkerson's algorithm $O(n^3)$.

Overall, the runtime complexity of the provided approach is $O(n^3)$.