Theory Assignment-5: ADA Winter-2024

Aarav Mathur (2022005) Aniket Gupta (2022073)

23-04-2024

1 Building the Flow-Network

Suppose that each box represents two nodes in our flow-network. Our goal is to construct a bipartite diagraph having vertex set $U \cup V$ such that each box is a vertex in both sets U and V, and there is an edge from $u \in U$ to $v \in V$ if and only if box u can be fitted inside box v.
We can represent this network-flow digraph using an adjacency matrix having $2n$ vertices. For any pair of boxes u and v, the adjacency matrix adj $(u, v \in \{1, 2, \ldots, n\})$ is defined as follows:

$$\text{adj}[u][v+n] = \begin{cases} 1 & \text{if box } u \text{ can be fitted inside box } v \\ 0 & \text{otherwise} \end{cases}$$

Thus we can clearly see that if $u \in U$, then $u \in \{1, 2, \ldots, n\}$ and if $v \in V$, then $v \in \{n+1, n+2, \ldots, 2n\}$. Determining whether box u can be fitted inside box v is straightforward. We can sort the dimensions of both boxes, which were initially denoted as $u.x$, $u.y$, $u.z$ for box u and $v.x$, $v.y$, $v.z$ for box v, since in the problem we are given that we are allowed to rotate the boxes. Then, we iterate through each dimension and check if $u.i < v.i$ for all i (where $i \in \{x, y, z\}$). Now, in order to complete our flow-network, we add source and sink vertices s, t to the graph, such that there is an edge from s to each vertex $u \in U$ and from each vertex $v \in V$ to t. Please note that in order to implement this, you would have to increase the size of our graph so that it contains 2 more vertices. s and t can be assigned with any index for the sake of inserting into our adjacency matrix. For simplicity, we would refer to these indices as s and t. Now,

$$\text{adj}[s][u] = \begin{cases} 1 & \text{if } u \in \{1, 2, \ldots, n\} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{adj}[v][t] = \begin{cases} 1 & \text{if } v \in \{n+1, n+2, \ldots, 2n\} \\ 0 & \text{otherwise} \end{cases}$$

We assign a capacity function $c : E \to \mathbb{Z}$ and $c(e) \geq 0$. Initialize c(e) = 1 for each directed edge present in our graph. Now we just need to push flow through this network using a max-flow algorithm like Ford-Fulkerson.

2 Why does maximum flow of the constructed network correspond to the answer of this problem?

We know from the applications of network flows that the maximum bipartite matching in the case of an unweighted undirected (which we convert to directed in the next step) bipartite graph equals the value of the maximum flow $|f|$ (can be directly referenced from the lecture slides). Take note that in the way we built the graph, we did not have to build an undirected graph but already achieved a directed graph. Also, the edges from the source s to vertex $u \in U$ are assigned with capactiy equal to one to enforce that one box cannot contain more than one boxes side-to-side. Now, if box u contains box v, select $e = (u, v+n)$ in our graph where $u \in U$ and $v \in V$. We know that each box can strictly contain only one other box "directly" inside it (i.e., no more than one box can be present inside one box kept side-to-side). Each time we place one box into another, the number of visible boxes gets decreased by one. For any legal nesting of the boxes, we know that it would correspond to a matching in our graph, and we need to find out the maximum matching. For any matching M, for any edge $(u, v+n)$ in our graph, the smallest dimension of box u is strictly smaller than the smallest dimension of box v, and thus, M

does not lead to a cycle of nested boxes, and corresponds to a valid nesting. Since the number of visible boxes is the number of boxes involved in a matching subtracted from the total number of boxes, the minimum number of visible boxes correspond to $n - |M'|$ where M' is the maximum matching, which in turn is equal to $n - |f'|$ where f' is the maximum flow (which we can compute using Ford Fulkerson's algorithm on our flow-network) subtracted from the total number of boxes.

3 Time Complexity

3.1 Building Flow-Network

- It takes $O((2n)^2)$ time to initialize the adjacency matrix.
- It takes $O(1)$ time to sort the dimensions of the boxes and make comparisons among 2 boxes since there is a constant number of dimensions (3).
- It takes $O(n^2)$ time to make edges from vertex u to vertex $v + n$ (the sorting done above is in $O(1)$ time) since we need to run a nested iteration to check if box u can be fitted inside box v.
- It takes $O(1)$ time to create vertices s and t.
- It takes $O(n)$ time to assign edges from s to U and from V to t.
- It takes $O(n^2)$ time to assign the capacity function $c : E \to \mathbb{Z}$ if number of edges $= O(n^2)$.
- Thus it takes $O(n^2)$ time in the worst-case to build the flow-network.

3.2 Applying Ford-Fulkerson's Algorithm to the Network

We are given that Ford-Fulkerson's algorithm runs in $O(|f|(|V| + |E|))$ time. $|V| = n$ and since the maximum possible value of $|f|$ is n, if the number of edges in our graph is $O(n^2)$, the worst-case time complexity of applying Ford-Fulkerson's algorithm to our flow-network becomes $O(n^3)$.

3.3 Overall Worst-Case Time Complexity

We have seen that building the flow-network takes $O(n^2)$ time, and applying Ford-Fulkerson's algorithm to the network takes $O(n^3)$ time. Thus, the overall time complexity of our algorithm is $O(n^3)$.