Theory Assignment-5: ADA Winter-2024

Vivaswan Nawani (2021217) Animesh Pareek (2021131)

1 Preliminaries

- We are given a Set of boxes B.
- In this Set of boxes B, every box has three dimensional geometry which is expressed in terms of height , width and length.
- It is also given that these dimensions (in centimeters) strictly belong to the range (10,20).
- We need to identify the minimum number of visible boxes.
- A box is visible only if it is not kept inside any other box.
- A box can only be placed in another box if and only if its length, width and height are smaller than the length,width and height of the holder box.
- This means that if height of the placed box exceeds the height of the holder box, then in that case placed box is counted as a visible box.

2 Problem-PreProcessing

- Let us start by first understanding this interesting constraint of dimension size.
- We are given, $10 < l, w, h < 20$ where l,w,h are length , width and height (in cms) of any box x ın B.
- This constraint on B in fact leads to an important observation, i.e, "given any two boxes belonging to B, we can never fit them together in any other box b which also lies in B".
- Thus, every box has only 2 choices:
  - Hold a single nested box tree
  - Hold no box
- Please Note that by Nested box tree, we mean the chain of non-visible boxes stacked inside each other.
- Based on the second last preliminary, we can infer that a placed box cannot be of same dimension as its holder box.
- Any box $b \in B$, with (l,w,h) as length, width and height of the box, can have only three possible rotation configurations:
  - Height based Rotation, $NEW\_HEIGHT = h$ (b_h)
  - Length based Rotation, $NEW\_HEIGHT = l$ (b_l)
  - Width based Rotation , $NEW\_HEIGHT = w$ (b_w)

3 Network-Flow Formulation

- We need only one Network-Flow graph G for this question.
- G starts with a source s and it ends at a sink t.
- In between we have 2 sets of vertices.
- First set is of holders, these are represented by h. We have one holder vertex per box in set B.
- While the Second set is of place occupiers (off course they are boxes), these are represented by p. We have a place occupier vertices per box in set B.
- Now we construct directed edges from source to each of the h vertices each having capacity 1.
- Next we need to draw edges holders and place occupiers.
- To do this we follow the following regime:
  - all edges are supposed to be directed from holders to place occupiers.
  - Edges connected the holders and place occupiers representing the same boxes are given a capacity of 0.
  - Remaining edges are given a capacity of 1 if the holder vertex of the edge can hold the place occupier vertex, else a capacity of 0 is assigned.
  - To check if h can hold p, either of these three conditions should be meet:
    - $HEIGHT(h_h) > HEIGHT(p_h)$ && either of the two:
      - $LENGTH(h_h) > LENGTH(p_h)$ && $WIDTH(h_h) > WIDTH(p_h)$
      - $LENGTH(h_h) > WIDTH(p_h)$ && $WIDTH(h_h) > LENGTH(p_h)$
    - $HEIGHT(h_h) > HEIGHT(p_l)$ && either of the two:
      - $LENGTH(h_h) > LENGTH(p_l)$ && $WIDTH(h_h) > WIDTH(p_l)$
      - $LENGTH(h_h) > WIDTH(p_l)$ && $WIDTH(h_h) > LENGTH(p_l)$
    - $HEIGHT(h_h) > HEIGHT(p_w)$ && either of the two:
      - $LENGTH(h_h) > LENGTH(p_w)$ && $WIDTH(h_h) > WIDTH(p_w)$
      - $LENGTH(h_h) > WIDTH(p_w)$ && $WIDTH(h_h) > LENGTH(p_w)$
- The above regime is encapsulated by our algorithmic function $fit(h, p)$
- Lastly we construct directed edges from each of the p to the sink each having capacity 1.
- The maximum flow of this Network-Flow Graph G represents the invisible boxes.

4 The specific Algorithm that solve the actual problem

To solve the actual problem, we follow the following steps:

- Step 0: Defining of $fit(h, p)$ as per network flow regime.
- Step 1: Taking up of the input of boxes(l,w,h) [SIZE = n]
- Step 2: Construction of the Network Flow Graph G
- Step 3: Calculating the Max flow of G using Ford Fulkerson's Algorithm
- Step 4: (Calculating output) $out = n - maxflow$

Fit algorithim:

- Input: Holder Boxes h and Place holder Box. Each box is defined by its length, width and height. (l,w,h)
- Check if h and p are same. ($h.l == p.l$ && $h.w = p.w$ && $h.h = p.h$)
- If found same the return 0
- If ($h.l > p.l$ && $h.w > p.w$ && $h.h > p.h$) return 1
- If ($h.l > p.w$ && $h.w > p.l$ && $h.h > p.h$) return 1
- If ($h.l > p.h$ && $h.w > p.w$ && $h.h > p.l$) return 1
- If ($h.l > p.w$ && $h.w > p.h$ && $h.h > p.l$) return 1
- If ($h.l > p.l$ && $h.w > p.h$ && $h.h > p.w$) return 1
- If ($h.l > p.h$ && $h.w > p.l$ && $h.h > p.w$) return 1
- if none of the above satisfies then return 0

Our main algorithm is as follows:

- Input: Set of Boxes B of size N, where each box is defined by its length, width and height. (l,w,h)
- Build Graph G as described in Network Flow Formulation. The capacity of the edges that are from holders to place occupiers in the graph is decided via the fit function. ($fit(h, p)$)
- The Graph G contains $(2 + 2N)$ vertices and $(N + N(N - 1) + N)$ edges.
- $Invisible\_boxes = Find\_maxflow(G)$
- Apply the Ford Fulkerson's Algorithm on this graph G to compute its max flow as follows:
- $Find\_maxflow(G)$
  - s= source of network flow graph G
  - t= sink of network flow graph G
  - $maxflow = 0$
  - $path$ = path from s to t using DFS (Depth First Search)
  - while ($path$!= NULL): while there is a path $path$ from s to t
    - f = minimum capacity of the $path$
    - subtract f from capacity of each edge in $path$
    - if capacity of any edge falls to 0 (or is less than equal to 0) remove that edge from the graph
    - $maxflow$ += f
    - $path$ = path from s to t using DFS (Depth First Search)
  - return $maxflow$
- $Visible\_boxes = N$ - $Invisible\_boxes$
- Output: $Visible\_boxes$

5 Complexity Analysis

- Time to construct graph:
  - Connecting source: $O(N)$
  - Connecting sink: $O(N)$
  - $fit(h, p)$ calling time: $O(1)$
  - Connecting holders and place occupiers: $O(Nx(N-1)) = O(N^2)$
  - Total: $O(N) + O(N^2) + O(N) = O(N^2)$
- Time of running Ford Fulkerson's Algorithm:
  - Max flow possible: N
  - vertices : $2 + 2N$
  - edges : $N + N(N-1) + N$
  - Time complexity : $O(maxflowx(|V| + |E|))$
  - Computed time : $O(N^3)$
- Time of computing output from $maxflow$: $O(1)$
- Total : $O(N^3)$

Also, in our algorithm the place where variable space is used in some order of N is in Graph construction and Path calculation. In both the cases we have space utilization in terms of edge and vertices storage. Edge Storage is in terms of N^2, while vertices is in term of N. Thus, Space complexity is in terms of N^2.
Thus, we have,

- Overall Time complexity : $O(N^3)$
- Overall Space complexity : $O(N^2)$

6 Proof of Correctness

- Given a graph which has a source s and a sink t, the Ford Fulkerson algorithm gives the maximum flow possible from s to t.
- This algorithm can be used to solve the classic maximum bipartite matching problem. In the maximum bipartite matching problem, we have a two disjoint set and independent set of vertices U and V such that every edge connects a vertex in U to a vertex in V. A possible valid matching is one, where no edge shares the same vertices, or in other words the mapping is one-one.
- The way we solve, the maximum bipartite matching problem is by taking a source and sink such that, all vertices in U are connected to the source and all vertices in V are connected to the sink. All edges have capacity 1.
- All edges having capacity of 1 ensures that in any path from source to sink, the minimum capacity available is either 0 or 1, so either an edge is taken(flow of 1) or not taken(flow of 0).
- Ford Fulkerson algorithm maximizes the flow, and thus maximizes the number of paths from source to sink, or in other words it creates a valid matching with maximum number of edges.
- We solve our original problem by reducing it to a problem of maximum bipartite matching. We can perform this reduction in polynomial time $O(N^2)$
- We need to minimize the number of visible boxes, or in other words we need to maximize the number of invisible boxes. One box can contain only one other box directly (without nesting) and one box can be contained by only one other box directly (without nesting)

- Thus, we create a set of vertices from the given boxes representing the holder boxes(U) and another set of vertices representing the place occupiers(V) and construct a graph with them. An edge from u, $u \in U$ to v, $v \in V$ indicates that u holds v with capacity 0(cannot hold) or 1(can hold). So, initially we construct the graph by checking if box u can fit box v and assigning it 0 or 1 accordingly (Note, a box cannot fit itself, and is hence assigned 0).
- Now our problem reduces to finding the maximum flow from U to V, which would give us the maximum number of invisible boxes. On subtracting that from the total number of boxes, we shall get the total minimum number of visible boxes.
- We have already shown earlier that the Ford Fulkerson algorithm correctly calculates the maximum flow in any graph form source to sink, and in the case of maximum bipartite matching it correctly finds the maximum bipartite matching, since we are reducing our original problem to the maximum bipartite matching problem, we can solve it correctly using the modified Ford Fulkerson algorithm as shown above..
- Hence Proved.