New CS 473 Spring 2015
Homework 9 Problem 2

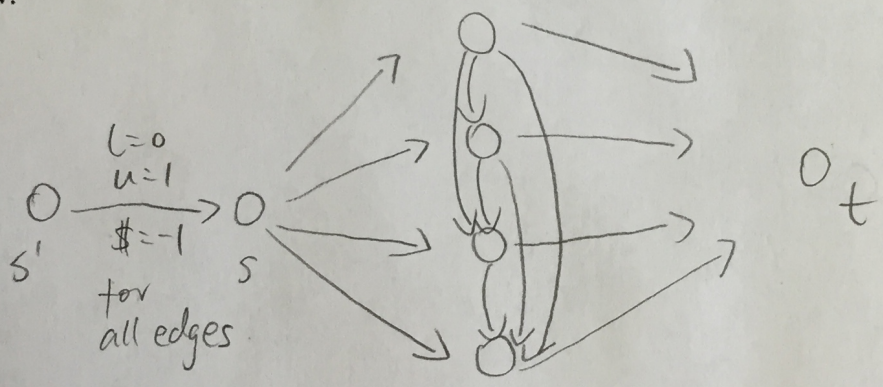
Arash Khatibi (khatibi2)
Liang Tao (ltao3)
Chen Zhang (czhang49)

1 Part A

The idea is to build a min cost flow problem , such that by solving for the min cost flow of the built graph G, we have the largest subset of the given boxes so that only one box is available. First we need to transform the boxes to a directed graph. Graph G can be built as follows: We represent each box as a node in the graph, and draw a directed edge from u to v if box u can be put into box v. Then put a directed edge from source to every node and a directed edge from every node to the target. Now basically the problem is to find the longest path in the directed graph G. We can do this by:

1. Assign the lower bound capacity $l(e)$ of every edge to 0 and the upper bound capacity of every edge to 1.
2. Assign the cost of every edge in the graph to -1.
3. Somehow limit the max flow in the graph to be 1.

Step 3 can be achieved by creating a new source node s' and put a directed edge from s' to s (note that the node s is no longer the source node of the graph). An example graph is illustrated as below:



This method works because by restricting the max flow of the Graph to be 1 (by the directed edge from s' to s), there can only be one path in the graph. And since the coast of every edge is -1, by solving for the min cost flow, we will have the longest path in the graph. To solve for the min cost flow, we can use the standard method or Orlin's method.

2 Part B

The idea is also to build a min cost flow problem , such that by solving for the min cost flow of the built graph G, we will know how to next the all the boxes so that the number of visible boxes is minimum.

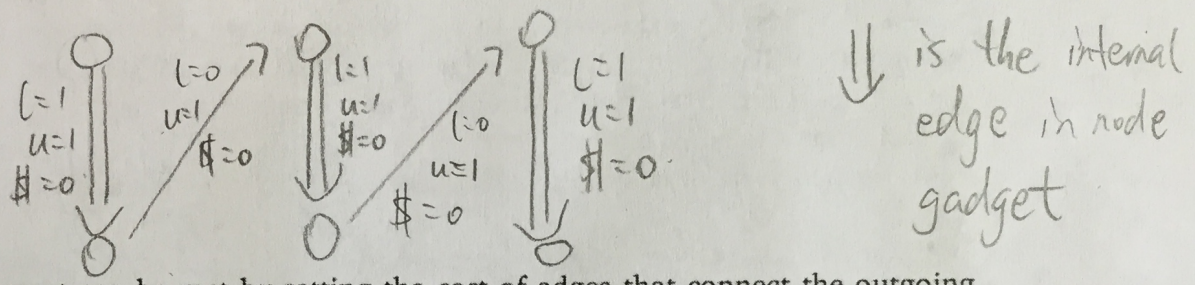
By analyzing the problem we know that there are two requirement we need to satisfy:

1. All the boxes need to be nested.
2. The number of visible boxes need to be minimum.

First we need to transform the boxes to a directed graph. Graph G can be built as follows (similar

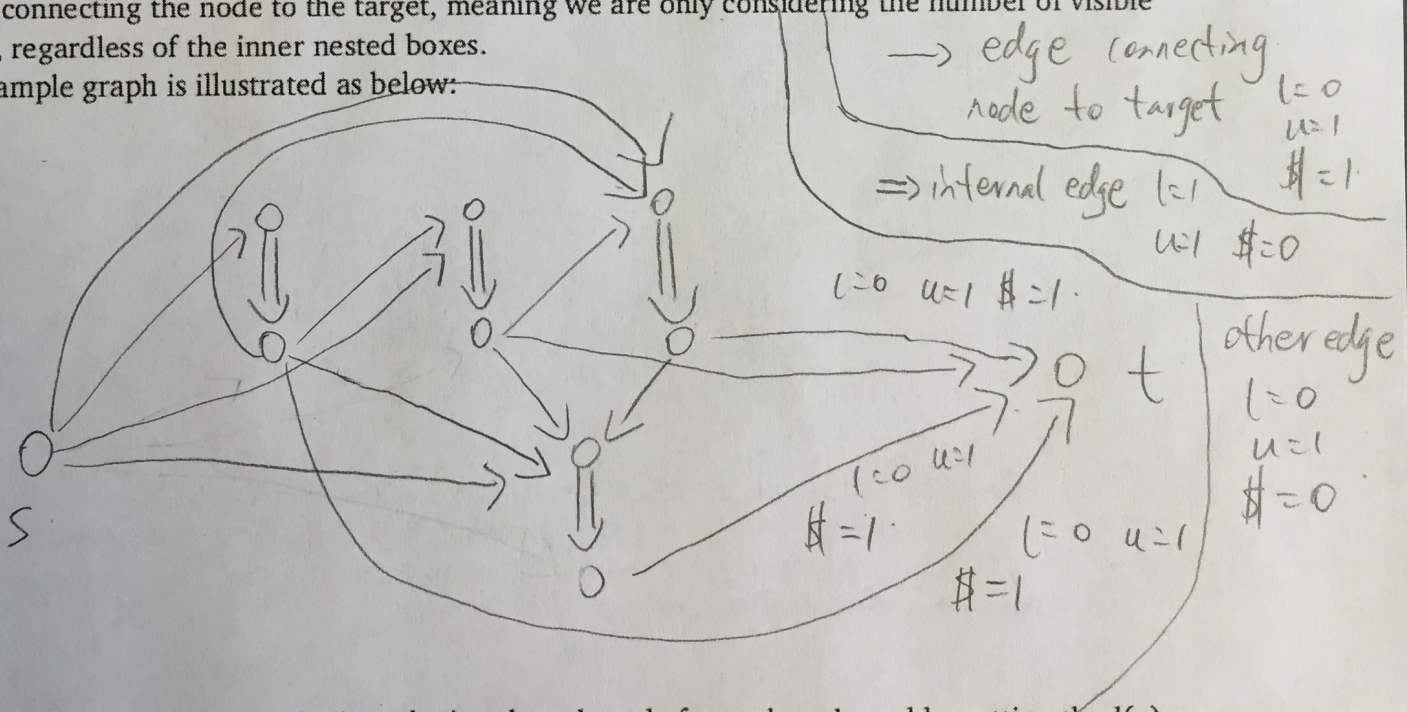
to Part A): We represent each box as a node in the graph, and draw a directed edge from u to v if box u can be put into box v. Then put a directed edge from source to every node and a directed edge from every node to the target. Note that if there is flow in the edge connecting the node to the target, it means that the box that the node represents is visible. Set the capacity of all edges to be $l(e) = 0$ and $u(e) = 1$.

Then the 1st requirement can be met by introducing the following gadget on every node. The idea is to expand each node into two sub-nodes, one sub-node for the incoming edges and one sub-node for the outgoing edge. Then connect the incoming sub-node to the outgoing sub-node and put the lower bound $l(e)$ and upper bound $u(e)$ of capacity of the connecting edge both to be 1. This guarantees that the flow through each node (representing of box) is 1, meaning that each box is only nested only once. An illustration is as below :



The second requirement can be met by setting the cost of edges that connect the outgoing sub-node of every node to be 1 and setting the cost of all other edges to be 0. This guarantees that no matter how the boxes are nested inside each other, the cost will only be affected by the edges connecting the node to the target, meaning we are only considering the number of visible boxes, regardless of the inner nested boxes.

An example graph is illustrated as below:



This method works because by introducing the sub-node for each node and by setting the $l(e)$ and $u(e)$ of the inner edge to be 1, we ensure that the flow through each node is 1 meaning that each box is either nested into other box or taken as a visible box. And since only the cost of the edges connecting the node to the target is 1 (if there is flow in the edge connecting the node to the target, it means that the box is visible), by solving for the min cost flow problem, we will know what is the minimum number of visible boxes.