CS 6820
HW 2
Question 4

Given the inputs, we will construct a graph, get the max flow from it, and get the schedule from the max flow, which will always be admissible. We will denote the total number of days in a given problem as d.

A note: this was discussed in lecture, but we define once again that for a directed edge (u, v), u is the tail and v is the head.

Graph Construction

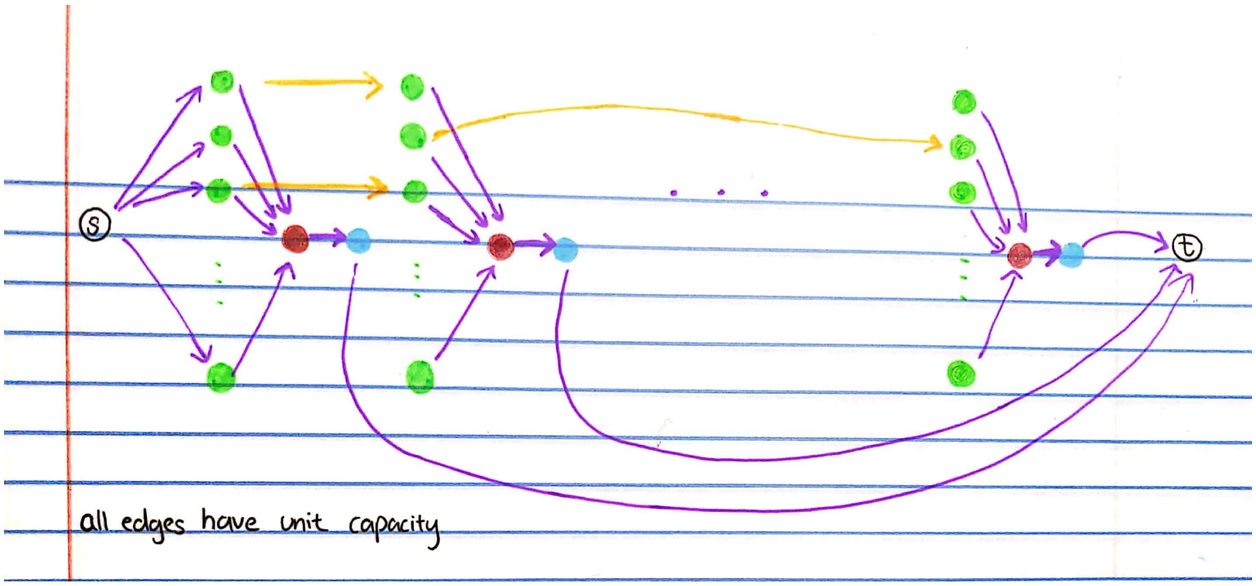

Figure 1: example flow network

Nodes

- Source node s
- Sink node t
- nd "town nodes" (green); every town has a node everyday
- d "day1 nodes" (red); one for each day
- d "day2 nodes" (blue); one for each day

Edges

- All edges have unit capacity
- For each day, there's an edge from the day1 to the day2 node

- For each day, there's an edge from the day2 node to the sink t
- For each day, there's an edge from every town node for that day to the day1 node
- "town-town" edges, who are generated by the following algorithm
- For the tail u of each "town-town" edge, have a (s, u) edge (recall s is the source)

Town-Town Edges

Before giving an exact algorithm/pseudocode, we discuss the general idea behind town-town edges. Each town-town edge goes from an earlier day to a later day connecting the same town; town-town edges never connect different towns. A town-town edge only exists if the garbage between those two days (inclusive) is less than 1. In every town τ between the endpoints of a town-town edge (u, v), there's the town-town edge (u, τ). Heads of town-town edges cannot be tails of town-town edges; if we term town-town edges as "intervals", then towns at the middle of an interval cannot be the starts of intervals, and ends of intervals also cannot be starts of intervals. The following is the pseudocode for constructing town-town edges.

```
function construct_town-town_edges(n, d,
garbage_information) {
  for town τ ∈ [1,...,n]:
    i = 1
    while day i < d:
      add edge (s,(τ,i))

      find the latest possible day j >= i that makes the
        garbage between day i and day j (inclusive) in town
        τ less than 1

      add edge ((τ,i), (τ,j))

      for day k ∈ [i+1,...,j-1]:
        add edge ((τ,i), (τ,k))

      i = j+1
}
```

The Algorithm

We run a max flow algorithm on this constructed flow network from s to t. To get the garbage collection schedule, we trace each saturated path P from s to t. For the last town node (τ, i) on P (i.e. the node right before going to the day1 node), we collect garbage on day i in town τ.

Time Complexity

Constructing the graph is polynomial time. There are $2 + nd + 2d$ nodes and $nd + 2d$ non town-town edges. The time complexity for the algorithm constructing town-town edges is $O(nd^2)$.

The max flow algorithms we've learned are all polynomial time.

Constructing the schedule from the max flow is also polynomial time because finding all saturated paths using DFS is polynomial time, $O(V + E)$, and finding the last town node on the path is constant time (fourth to last node).

Proof of Correctness

For each town-town edge (recall we termed them "intervals"), the source s is connected to the start of the interval and the end of the interval is connected to a day1 node, which is then connected to a day2 node, which finally is connected to the sink t. **The max flow is thus exactly the number of unique interval starts.** Our algorithm procures an admissible schedule because first of all, the truck never visits more than one town per day because each day1 node has unit outgoing capacity (as does every edge in the graph), restricting the truck to only visit one town per day; and second of all, for each town's saturated intervals, the truck visits at the end of the interval, and for each pair of adjacent intervals in a town, the most extreme case is when the truck visits at the very beginning of the first one and at the very end of the second one, but even in that case, the garbage buildup is < 2 before the second visit because each of the two intervals' garbage buildup is < 1.