Alex Havrilla

Hwk 1

Task 1:

Construct a bipartite graph G = (V,E), $V = C \cup P, C \cap P = \emptyset$ where $u \in C$ corresponds to chapters and $v \in P$ correspond to problems. Put an edge infite capacity between $u \in C, v \in P$ if reading the chapter u is required to solve problem v. Now connect a source vertex s to vertices $u \in C$ whose edges have capacity equal to the cost it takes to read the chapter u. Similarly attach a sink vertex t to $v \in P$ whose capacities are the value of the problem v.

We now argue a correspondence betweeen cuts produced by flows and groupings of chapters read and problems not done. Suppose we have a cut. It cannot contain any edges between C and P since these edges have infinite capacity. We form our grouping by considering those chapters not in the partition with s and those problems not in the partition with t. The sum of the capacities of these cut edges is then the sum of the cost of the chapters read and the sum of the value of the chapters not done. Now consider a valid grouping of chapters read and problems not done. Form the cut by cutting the edges of those chapters adjacent to the source which have been read the cutting the edges of those problems adjacent to the sink which are not done. We show this is a valid cut by showing every path is has a cut edge. We know every path goes from the source to a chapter to a problem to sink. Assume that for some path the edge to the chapter is not cut. Then we know this chapter has not been read. So the connected problem could not have been solved and its edge is therefore cut. Since the path was arbitrary this demonstrates a valid cut, allowing us to conclude the desired bijection.

Further we know that the capacity of a cut is the sum of the read chapter costs plus the sum of the undone problems. We can compute the maximal utility as the sum of the finished problem values minus the sum of the chapter costs. But this can be rewritten as the total sum of the chapters minus the

capacity of a cut. Thus minimizing the capacity of a cut maximizes the utility.

This cut can be computed in polynomial time using EK2 or a variant, from which we compute the chapters to bread and problems to be solved(and the max utitility).

Task 2:

---

a)

$$R_{tw} = \begin{cases} \frac{w+S}{min(S,t+E)} & w < t \\ \frac{t+E}{min(S,t+E)} & w \geq t \end{cases}$$

b)

We show $R_{tw} \leq R_{\infty w}$ for $t \neq \infty$. Compute

$$R_{\infty w} = \frac{w+S}{S} = \frac{w+3}{3}$$

since the elevator will never arrive. We computed above

$$R_{tw} = \begin{cases} \frac{w+S}{min(S,t+E)} & w \leq t \\ \frac{t+E}{min(S,t+E)} & w > t \end{cases}$$

In the first case estimate since $t \geq 2$ we know $min(S, t+E) = 3$

$$\frac{w+S}{min(S,t+E)} = \frac{w+3}{min(3,t+E)} \leq \frac{w+3}{3} = R_{\infty w}$$

In the other case $t < w$ so

$$\frac{t+E}{min(S,t+E)} \leq \frac{w+1}{3} < \frac{w+3}{3} = R_{\infty w}$$

This formally justifies the adversary should only ever choose $t = 1$ or $t = \infty$ since in every case for fixed w picking otherwise is guaranteed to result in lower payoff.

c)

Formally we show for $w > 1$, $R_{tw} \geq R_{t1}$ for $t \in \{1, \infty\}$. Compute for $t = \infty$

$$R_{\infty w} = \frac{w+S}{S} = \frac{w+3}{3} \geq \frac{1+3}{3} = R_{t1}$$

for $w > 1$

Compute for $t = 1$

$$R_{1w} = \frac{t+E}{min(S, t+E)} = \frac{2}{2} = 1 \geq R_{11} = 1$$

This shows the column player should(waiter) should never wait for more than 1 unit as for arbitrary t this is guaranteed to be worse than waiting for 0 or 1 unit.

d)

Compute $R_{1,0} = \frac{S}{1+E} = \frac{3}{2}$, $R_{1,1} = 1$, $R_{\infty,0} = 1$, $R_{\infty,1} = \frac{1+S}{S} = \frac{4}{3}$. This gives the matrix

$$\begin{bmatrix} \frac{3}{2} & 1 \\ 1 & \frac{4}{3} \end{bmatrix}$$

Suppose the column player takes $w = 0$ with probability p(and $w = 1$ with $1 - p$). It suffices to consider only pure row strategies. The payoff of the row player should they always take $t = 1$ is then $\frac{3}{2}p + 1 - p$. Should they take $t = \infty$ this has expected payoff $v - (p_1 + \frac{4}{3}p_2)$.

The corresponding LP has variables v, p_1, p_2 which is seeking to maximize v subject to constraints $p_1, p_2, v \geq 0$, $p_1 + p_2 \leq 1$ and $v - (\frac{3}{2}p_1 + p_2) \leq 0$, $v - (p_1 + \frac{4}{3}p_2) \leq 0$.

We seek to compute $max_{p \in [0,1]} min(\frac{3}{2}p_1 + 1 - p_1, p_1 + \frac{4}{3}(1 - p_1))$. Note that the left expression(as a function f_1 of p_1) is increasing while the right f_2 is decreasing. Further $f_1(0) = 1$ while $f_1(1) = 3/2$. While $f_2(0) = 4/3$ and $f_2(1) = 1$. Via IVT they intersect for some p. Thus we know the maximizer of the minima occurs at this intersection. Solving gives $p_1 = 2/5 \implies p_2 = 3/5$. The value of the game is then $6/5$.

e)

The row player is seeking to maximize their minimum score(via von Neumann minimax theorem). It suffices to consider pure strategies of the column player. If the col player takes $w = 0$ the expected payoff of the row player is $\frac{3}{2}q + 1 - q$. But we know the value of the game is $6/5$ which will occur for at the intersection of this expression and the decreasing expression for the case $w = 1$. So it suffices to solve for q when $3/2q + 1 - q = 6/5$, yielding $2/5$. Hence the row player should pick $t = 1$ $2/5$ of the time and $t = \infty$ $3/5$ of the time.

Task 3:

---

a)

The optimal solution is one with total length 113. This is achieved by setting the length of 0 to 88, 99 to 13, 100 to 12, and 0 for the rest.

b)

Formally let $P, N \subseteq [n]$ with $P \cup N = [n], P \cap N = \emptyset$. Our variables are $l_1, ..., l_n$ representing the lengths of wires attached to nodes $v_1, ..., v_n$. We seek to minimize $c^T l$ where $c = [1]^n$ with contraints s.t. for all $i \in P, j \in N$, $l_i + l_j \geq d_{i,j}$ where $d_{i,j}$ is the euclidean distance between v_i and v_j. Additionally we constrain $l_i \geq 0$ for $i \in [n]$.

For $P = 1, 2$ and $N = 3, 4$ we minimize

$$l_1 + l_2 + l_3 + l_4$$

subject to

$$\begin{cases} l_1 + l_3 \geq d_{1,3} & l_1 + l_4 \geq d_{1,4} \\ l_2 + l_3 \geq d_{2,3} & l_2 + l_4 \geq d_{2,4} \\ l_1 \geq 0 & l_2 \geq 0 \\ l_3 \geq 0 & l_4 \geq 0 \end{cases}$$

c)

In the dual our variables are $y_{i,j}$ for each pair $i \in P, j \in N$. We seek to maximize

$$\sum_{i \in P, j \in N} d_{i,j} y_{i,j}$$

subject to the constraints $y_{i,j} \geq 0$ and

$$\begin{cases} \forall i \in P, \sum_{j \in N} y_{i,j} \leq 1 \\ \forall j \in N, \sum_{i \in P} y_{i,j} \leq 1 \end{cases}$$

With respect to the example $P = \{1, 2\}, N = \{3, 4\}$ we have variables $y_{1,3}, y_{1,4}, y_{2,3}, y_{2,4}$ trying to maximize

$$d_{1,3} y_{1,3} + d_{1,4} y_{1,4} + d_{2,3} y_{2,3} + d_{2,4} y_{2,4}$$

subject to

$$\begin{cases} y_{1,3}, y_{1,4}, y_{2,3}, y_{2,4} \geq 0 & \\ y_{1,3} + y_{1,4} \leq 1 & y_{2,3} + y_{2,4} \leq 1 \\ y_{1,3} + y_{2,3} \leq 1 & y_{1,4} + y_{2,4} \leq 1 \end{cases}$$

d)

We can solve the dual using a mincost-maxflow algorithm. Construct a complete bipartite graph of size n with nodes corresponding to terminals and the partition given by P, N. Let the capacities of the edges be 1 and their costs be $-d_{i,j}$. Attach a source vertex to nodes in P whose edges have unit capacity and 0 cost. Similarly attach a sink vertex to nodes in N whose edges have capacity 1 and cost 0. We now argue a correspondence between valid assignments to $y_{i,j}$ and flows. Suppose we have a flow M on our graph. Then to produce a satisfying assignment we set $y_{i,j}$ to be the flow through vertex v_i to vertex v_j. We are guaranteed for each $i \in P$, $\sum_{j \in N} y_{i,j} \leq 1$ since the inflow to node v_i is at most 1 and hence the outflow, ie. the sum of

the $y_{i,j}$ is 0. We may run a symmetric argument for the negative constraints. Now suppose we have a valid assignment. Then we produce a flow by setting the in flow of each $v_i, i \in P$ to be the sum $\sum_{j \in N} j_{i,j}$ and the flow from v_i to $v_j, j \in N$ to be $y_{i,j}$. The positive constraints ensure the capacity constraints are not violated for the positive vertex inflows and the negative constraints ensure the negative vertex outflow capacities are not violate. Hence we have a valid flow. This establishes a bijection.

Then running a mincost maxflow algorithm finds us a minimizer of the sum of the negative distances which corresponds to a maximizer of the sum of the positive distances. Note that the flows correspond to tightness on some of the constraints, which is where a maximizer must occur.