CS535 Fall 2022 HW4 Sample Solutions

1. By Menger's theorem, after deleting any $k-1$ edges in the graph, you can neither disconnect u from v, nor disconnect v from w, there must exist a $u-v$ path and a $v-w$ path, therefore a $u-w$ path. Since by deleting any $k-1$ edges you cannot disconnect u and w, they are k-edge-connected.

2. Greedy algorithm. Start with $i=1$ and $j=1$. As long as the remaining supply of s_i and remaining demand of t_j is not 0, ship as many cars from s_i to t_j as possible and record the cost. If the supply of s_i reaches 0, switch to $i+1$. If the demand of t_j reaches 0, switch to $j+1$. Each time at least 1 of i and j increases, the time complexity is $O(m+n)$.

   Sort all pairs (i,j) in lexicographical order. This algorithm assigns values to the variables in lexicographical order, greedily making each value as large as possible. Let x be the greedy solution, and x^* be the leximally optimal solution. We show that $x=x^*$. Denote $c_{ij}=p_i \vee q_j$.

   Proof by contradiction. Let (i,j) be the first edge such that $x_{ij} \neq x^*_{ij}$. By the greedy choice of x_{ij}, $x_{ij} > x^*_{ij}$. In order to ensure that the supply a_i and the demand b_j are both satisfied, x^* must assign positive values to at least two distinct variables x^*_{ik} and x^*_{lj} such that $i<l\leq m$ and $j<k\leq n$. Let $\varepsilon = \min\left\{x^*_{ik}, x^*_{lj}\right\}$. Since

   $$c_{ij}+c_{lk} \leq c_{ik}+c_{lj},$$

   reducing x^*_{ik} and x^*_{lj} by ε and increasing x^*_{ij} and x^*_{lk} by this same ε gives a new feasible solution y with the same or lower cost. Hence y is also optimal. However, y is the lexicographically greater than x^*, which contradicts to the choice of x^*.

3. Let $a=(u,v)$. In the residual graph compute the shortest path p from v to u in $O(nm)$. If $l(p)+l(a)<0$, push 1 unit of circulation on a and p and it's the new min-cost TS, otherwise the min-cost TS is the same as before.

4. Create a "sink" t and create an arc (v,t) for all $v\in V$ with 0 cost. Let $b(t)$ be the difference between the total supplies and total demand. Now it has been transformed to an ordinary min-cost TS problem.

5. Maximizing the revenue is equivalent to minimizing the negation of revenue, therefore we construct a flow network with the edge lengths being the negation of all the revenues.

   Create a source s with $b(s)=-m$.

   For each $1\leq i\leq n+1$, create a vertex v_i representing the beginning of each year i with $b(v_i)=0$.

   For each $1\leq i\leq n$, create vertex h_i representing harbor to store unused ships with $b(h_i)=0$.

   For each $1\leq i\leq n$, create vertex u_i, w_i with $b(u_i)=-d_i$ and $b(w_i)=d_i$.

   Create a sink t with $b(t)=m$.

   All created edges have default capacity $c=+\infty$ and demand $d=0$ and length $l=0$ unless specified otherwise. Create arc (s,v_1) and (v_{n+1},t).

   For each $1\leq i\leq n$, create arcs (v_i,h_i), (h_i,v_{i+1}) representing storing unused ships.

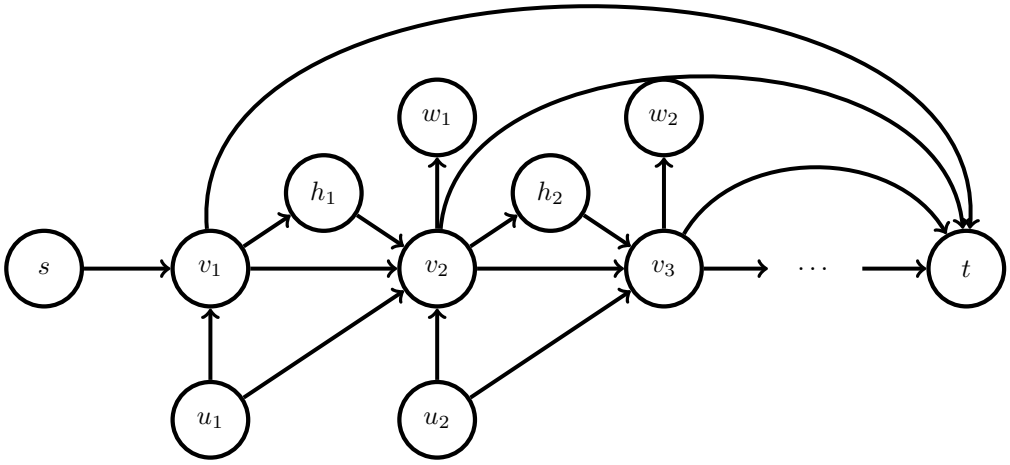
   For each $1\leq i\leq n$, create arc (v_i,t) with length $l=-s_i$ representing selling the ships.

   For each $1\leq i\leq n$, create arc (v_i,v_{i+1}) with demand $d=d_i$ and length $l=-r_i$ representing the revenue of fulfilling the market demand.

   For each $1\leq i\leq n$, create arc (u_i,v_i) with length $l=h_i+r_i$ representing hiring ships.

   For each $1\leq i\leq n$, create arc (u_i,v_{i+1}) and (v_{i+1},w_i).

   Now compute a feasible min-cost TS and the negation of its cost is the solution to the original problem.



6. (a) Prove by contradiction. Assume $\varphi_1 < \varphi_2$ and $mc(\varphi_1) \geq mc(\varphi_2)$. Let f_1 and f_2 be min-cost flows of value φ_1 and φ_2, respectively. Let $P = f_1 \oplus f_2$. P is a collection of $s-t$ paths. Let $P_1 = P \cap f_1$ and $P_2 = P \cap f_2$. By the assumption, the flow value of P_1 is smaller than P_2, yet the total cost of P_1 is higher than P_2. Since all paths have positive costs, we are able to get a decomposition P_2' from P_2 with the flow value of P_1, with a smaller cost than P_1. Replace P_1 by P_2' in f_1, we get a flow with lower cost, contradicting the fact that f_1 is min-cost. Therefore, $mc(\varphi)$ strictly increases with φ.

   (b) With (6a) proven, we can binary search between 0 and φ^* to get the largest φ with $mc(\varphi) \leq B$. Each min-cost flow computation is polynomial time, and we repeat $O(\log(\varphi^*))$ times. Let $c_{max} = \max_{a \in A} c(a)$. Since $c(a)$ are positive integers for all $a \in A$,

$$\varphi^* \leq m \cdot c_{max}$$
$$\log(\varphi^*) \leq \log(m \cdot c_{max}) \leq m \cdot \log(c_{max})$$

   Note that the input size of the capacity vector c is $O(m \cdot \log(c_{max}))$. Therefore $\log(\varphi^*)$ is polynomial with respect to the input size, the entire process can be done in polynomial time.