COMP 360 - Sample Midterm

1. Chemco produces two chemicals: A and B. These chemicals are produced via two manufacturing processes. Process 1 requires 2 hours of labor and 1kg of raw material to produce 1g of A and 1g of B. Process 2 requires 3 hours of labor and 2kg of raw material to produce 3g of A and 2g of B. Sixty hours of labor and 40kg of raw material are available. Chemical A sells for \$16 per gram and B sells for \$14 per gram. Formulate a linear program that maximizes Chemco's revenue.

   Solution: Let x_1 be the number of times we run process 1, and x_2 be the number of times we run process 2.

$$
\begin{array}{ll}
\max & (16+14)x_1 + (3 \times 16 + 2 \times 14)x_2 \\
\text{s.t.} & 2x_1 + 3x_2 \leq 60 \\
 & x_1 + 2x_2 \leq 40 \\
 & x_1 \geq 0 \\
 & x_2 \geq 0
\end{array}
$$

2. There are n athletes, numbered $1, \ldots, n$, and m races are scheduled between them. Every race is between five athletes and has one winner. More formally we are given m sets $R_1, \ldots, R_m \subseteq \{1, \ldots, n\}$ each of size exactly 5, where R_i is the list of the athletes in the i-th race. We are also given positive integers $p_1, \ldots, p_n$. We want to see if it is possible for the races to finish in such a way that the i-th athlete wins at most p_i races (for all $1 \leq i \leq n$). Show that this problem can be modeled as a max flow problem and solved using the Ford-Fulkerson algorithm.

   Solution: Create a node for each race, and one for each athlete, and direct 5 edges of capacity 1 from each race to the five athletes in that race. Connect the source to each race with an edge of capacity 1, for each i, connect the i-th athelete to the sink with an edge of capacity p_i. If Max-flow equals m, then we know that it is possible for the races to finish in such a way that the i-th athlete wins at most p_i races. Indeed in that case, in an integer valued flow, the edge with one unit of flow on it going of each race will determine the winner of that race.

3. Prove that for every flow network $(G, s, t, \{c_e\})$, there is a maximum flow f such that the edges with non-zero flow on them form a graph that does not contain any directed cycles.

   Solution: We first find a max-flow and then modify it to remove all the directed cycles. If there is such a cycle, we decrease the flow on all the edges of the cycle by the smallest flow value on the cycle, and we repeat this process until all the cycles are gone. This preserves the conservation condition because it decreases the same amount from the incoming and outgoing flow of the vertices on the cycle. Capacity condition is also obviously remains valid. Moreover the value of the flow does not change because the cycle cannot pass through the source s. So at every step we arrive at a new valid max flow. This process has to terminate because at every step at least one new edge will have 0 flow on it and the number of edges is finite. Finally when the algorithm terminates we will not have any cycle with positive flow on it as otherwise the algorithm would have not halted.

4. We are given two positive numbers n and d and a function $f : \{0, 1, \ldots, n\} \to \mathbb{R}$. Our goal is to find the best approximation of f with a polynomial of degree d. More precisely we want to find a polynomial of degree d that minimizes $\max_{x \in \{0,\ldots,n\}} |f(x) - p(x)|$.

   (a) Formulate this problem as a linear program.

   Solution: The variables are $a_0, \ldots, a_d, e$, and they are all free:

$$\begin{array}{llll} \min & e & & \\ \text{s.t.} & e + \sum_{i=0}^{d} k^i a_i & \geq f(x) & \forall k \in \{0, 1, ..., n\} \\ & e - \sum_{i=0}^{d} k^i a_i & \geq -f(x) & \forall k \in \{0, 1, ..., n\} \end{array}$$

   (b) Write the dual of your linear program.

   Solution: The variables are $b_0, \ldots, b_n, c_0, \ldots, c_n$:

$$\begin{array}{llll} \max & \sum_{k=0}^{n} f(k) b_k - f(k) c_k & & \\ \text{s.t.} & \sum_{k=0}^{n} k^i b_k - \sum_{k=0}^{n} k^i c_k & = 0 & \forall i \in \{0, 1, ..., d\} \\ & \sum_{k=0}^{n} b_k + c_k & = 1 & \\ & b_k, c_k & \geq 0 & \forall k \in \{0, 1, ..., n\} \end{array}$$