**Due 23:59 Nov 30 (Sunday)**. There are 100 points in this assignment.
Submit your answers **(must be typed)** in pdf file to CourSys
`https://coursys.sfu.ca/2025fa-cmpt-705-x1/`.
Submissions received after 23:59 will get penalty of reducing points: 20 and 50 points deductions for submissions received at $[00:00, 00:10]$ and $(00:10, 00:30]$ of Dec 1, respectively; no points will be given to submissions after $00:30$ of Dec 1.

1. (Chapter 10 Problem 1 of the text book) 15 points

   The Hitting Set problem is defined as follows: Let $A = \{a_1, .., a_n\}$ and $B_1, .., B_m$ be a collection of subsets of $A$, a subset $H$ of $A$ is a hitting set for $B_1, .., B_m$ is $H \cap B_i \neq \emptyset$ for every $1 \leq i \leq m$. The Hitting Set problem is that given $A$ and $B_1, .., B_m$, and integer $k > 0$, whether there is a hitting set of size $k$ or not. The problem is NP-complete. Assume that $|B_i| \leq c$ for some constant $c > 0$. Give an algorithm that solves the problem with a running time of $O(f(c, k) \cdot p(n, m))$, where $p(n, m)$ is a polynomial in $n$ and $m$, and $f(c, k)$ is an arbitrary function depending only on $c$ and $k$, not on $n$ or $m$, and analyze the algorithm.

   **Solution**

   We design a branching algorithm similar to the one used for the VERTEX COVER problem.

   Let $A = \{a_1, \ldots, a_n\}$ and let $B_1, \ldots, B_m$ be subsets of $A$ such that each $|B_i| \leq c$, where $c$ is a fixed constant. We want to determine whether there exists a hitting set $H \subseteq A$ of size $k$, i.e., a set $H$ such that $H \cap B_i \neq \emptyset$ for all $1 \leq i \leq m$.

   **Key idea.** If $H$ is a hitting set of size $k$, then for any set $B_1$, at least one element of $H$ must lie in $B_1$. Thus, for each $u \in B_1$, we can *branch* on choosing $u$ into the hitting set. If we choose $u$, then every set $B_i$ that contains $u$ is already hit and can be removed. We then recurse on the remaining sets and with budget $k - 1$.

   **Algorithm.**

   Let $\mathrm{HS}(B, k)$ denote the decision procedure.

$$\mathrm{HS}(B,k): \begin{cases} \text{return True}, & \text{if } B = \emptyset, \\ \text{return False}, & \text{if } k = 0 \text{ and } B \neq \emptyset, \\ \text{Pick any } B_1 \in B. & \\ \text{For each } u \in B_1: & \\ \qquad B' := \{\, B_i \in B : u \in B_i \,\} & \text{(sets hit by } u), \\ \qquad \text{if } \mathrm{HS}(B \setminus B',\ k-1) = \text{True, return True.} & \\ \text{return False.} & \end{cases}$$

**Correctness.** Every valid hitting set must contain at least one element of $B_1$. Thus the branching over all $u \in B_1$ preserves completeness. Removing the sets containing $u$ is valid because they are already hit by choosing $u$. Since we reduce $k$ by one at each recursive call, the recursion depth is at most $k$.

**Running time analysis.**

Since $|B_1| \leq c$, each recursive step creates at most $c$ branches. Thus the recursion tree has at most

$$c^k = O(f(c,k))$$

leaves. At each node, removing the sets containing $u$ takes polynomial time, say $p(n,m)$.

Therefore the total running time is

$$O(f(c,k)\,p(n,m)) = O(c^k \cdot p(n,m)),$$

where $p(n,m)$ is a polynomial and $f(c,k) = c^k$ depends only on $c$ and $k$.

**Conclusion.** Since $c$ is a constant, the algorithm runs in fixed-parameter tractable time $O(f(c,k)\,p(n,m))$, as required.

2. Chapter 10 Problem 8 of the text book) 15 points

Consider the class of 3-SAT instances in which each clause has exactly three literals and each of the $n$ variables appears (counting $x$ and $\bar{x}$) in exactly three clauses. Prove that every such 3-SAT instance is satisfiable and a satisfying assignment can be found in polynomial time.

**Solution**

There are $3n$ literals. Each clause has 3 literals. Thus $m = \frac{3n}{3} = n$ There are as many variables as clauses.

We want to match each clause with exactly one variable. Then, for each clause exactly one variable is set so that satisfies the clause.

I prove there is a perfect matching using [Frobenius 1917, Hall 1935] theorem in the lecture note.

Each set of x variables, is connected to 3x literals. That means at least x clauses.

Each set of y clauses is connected to 3y literals, that means at least y variables.

Thus there is a perfect matching from each varialbe to a clause.

The algorithm that returns the solution is network flow:

Connect one edge from a node for each variable to a node for each clause in which the literal related to that variable exist.

Connect source node s to each variable. connect each clause node to t.

set all capacities equal to 1. Number of edges are $3n + 2n$. Find the maximum flow using Ford-Falkerson algorithm in $O(n^2)$ time.

3. (Chapter 13 Problem 4 of text book) 10 points

   A peer-to-peer network (digraph) $G$ is constructed as follows:
   $V(G) = \{v_1\}$;
   for $i = 2$ to $n$ select randomly with same probability a node $v_j$ in $G$, $V(G) = V(G) \cup \{v_i\}$ and $E(G) = E(G) \cup \{(v_i, v_j)\}$;

   Give a formula on the expected indegree of each node $v_j$ in terms $n$ and $j$ for $1 \le j \le n$.

   **Solution:**
   $E(ind(v_i)) = \sum_{i=1}^{n-1} \frac{1}{i} = H_{n-1}$
   For the first round, no indegree will be added.
   For the second $v$, only 1 to $v_1$
   for the third round, 1 with prob 1/2 for $v_1$ and $v_2$
   for the jth round, 1 with prob $1/(j-1)$ for $1_1 tov_{j-1}$

   Taking summation of expectations of in-degree added to $v_i$ in each round we get the formula.

4. (Chapter 13 Problem 7 of the text book) 10 points

   Given a set of clauses $C_1, .., C_k$ of Boolean variables $X = \{x_1, .., x_n\}$ with each clause has at least one literal and the literals in single literal clauses are distinct, the MAX SAT problem is to find a truth assignment satisfying as many clause as possible. A randomized algorithm for the MAX SAT is to assign each $x_i$ independently to 0 or 1 with probability 1/2. Show that the expected number of clauses satisfied by this algorithm is at least $k/2$. Give an example to show that there are MAX SAT instances such that no assignment satisfies more than $k/2$ clauses.

   **solution:**
   if $c_i$ has $s_i$ literals:

   $Prob(C_i \text{ is satisfied}) = 1 - (\frac{1}{2})^s_i$

   $E(\text{ number of clauses satisfied}) = \sum_i Prob(C_i \text{ is satisfied}) = \sum_i (1 - (\frac{1}{2})^s_i) = k - \sum_i ((\frac{1}{2})^s_i)$

   since $s_i \ge 1$ :

   $k - \sum_i ((\frac{1}{2})^s_i) \ge k - \sum_i ((\frac{1}{2})) = k - k/2 = k/2$

   Example: 1 variable and $k = 2$ clauses:
   $(x_1)(\bar{x_1})$

   There are at most $k/2 = 2/2 = 1$ clauses satisfiable.

5. (Chapter 13 Problem 10 of text book) 10 points

   There are $n$ bidding agents; agent $i$ has a distinct integer bid $b_i > 0$. The bidding agents appear in an order chosen uniformly at random, each proposes its bid $b_i$ in turn, and at all times the system maintains a variable $b^*$ equal to the highest bid seen so far. (Initially $b^*$ is set to 0.) What is the expected number of times that $b^*$ is updated when this process is executed, as a function of $n$?

**solution:**
$Prob($ updated at round 1$) = pr(b_1 > 0) = 1$
$Prob($ updated at round 2$) = pr(b_2 > b_1) = 1/2$ the last equality is since each $b_i$ is identically distributed.
$Prob($ updated at round 3$) = pr(b_3 > b_1, b_3 > b_1) = pr(b_3 > b1) \times pr(b_3 < b_1) = 1/2 \times 1/2 = 1/4$ Since each $b_i$ is independent.

$Prob($ updated at round j$) = pr(b_j > b_1, b_2, ..., b_{j-1}) = \frac{1}{2^{j-1}}$

$E[\textbf{number of updates}] = \sum_{i=0}^{n-1} \frac{1}{2^i} = \frac{1-\frac{1}{2^n}}{1-\frac{1}{2}}$

6. (Chapter 13 Problem 11 of the text book) 15 points

   There are $k$ machines and $k$ jobs. Each job is assigned to one of the $k$ machines independently at random (with each machine equally likely).

   (a) Let $N(k)$ be the expected number of machines that do not receive any jobs, so that $N(k)/k$ is the expected fraction of machines with no job. What is the limit $\lim_{k\to\infty} N(k)/k$? Give a proof of your answer.

   **solution:**
   $pr($machine i does not receive a job$) = (\frac{k-1}{k})^k$
   $E($number of machines with no jobs$) = \sum_i (\frac{k-1}{k})^k = k \times (\frac{k-1}{k})^k$
   $\lim \frac{k\times(\frac{k-1}{k})^k}{k} = \lim(1-\frac{1}{k})^k = e^{-1}$

   (b) Suppose that machines are not able to queue up excess jobs, so if the random assignment of jobs to machines sends more than one job to a machine $M$, then $M$ will do the first of the jobs it receives and reject the rest. Let $R(k)$ be the expected number of rejected jobs; so $R(k)/k$ is the expected fraction of rejected jobs. What is $\lim_{k\to\infty} R(k)/k$? Give a proof of your answer.

   **solution:**
   as much as number machines with no jobs, jobs is going to machines with at least one job. This is because number of jobs and machines are equal. Thus the answer is similar to part a: $e^{-1}$

   (c) Now assume that machines have slightly larger buffers; each machine $M$ will do the first two jobs it receives, and reject any additional jobs. Let $R_2(k)$ denote the expected number of rejected jobs under this rule. What is $\lim_{k\to\infty} R_2(k)/k$? Give a proof of your answer.

   **solution:**
   Let $X_i$ be the number of jobs assigned to machine i: $E[$number of jobs machine i can process$] = 0\times prob(X_i = 0) + 1\times prob(X_i = 1) + 2\times prob(X_i \geq 2) = prob(X_i = 1) + 2\times(1 - prob(X_i =$

$0) - prob(X_i = 1)) = 2 - 2 \times prob(X_i = 0) - prob(X_i = 1)$

$R_2(K) = 2k - 2k \times (\frac{k-1}{k})^k - k \times (\frac{k-1}{k})^{k-1}$
$\lim R_2(K)/k = 2 - 3 \times e^{-1}$

7. 15 points

   Given a graph $G$, we consider the following problem: color the nodes of $G$ using $k > 1$ colors; an edge $e = \{u, v\}$ is called satisfied if $u$ and $v$ are colored by different colors; and we want to color the nodes to satisfy as many edges as possible. A randomized algorithm RA for the maximization problem is as follows: for every node $v$ of $G$, select one of the $k$ colors independently with probability $1/k$ and assign the color to $v$.

   (a) Prove that the expected number of edges satisfied by RA is $(1 - 1/k)m$, where $m$ is the number of edges in $G$. Assume $(1 - 1/k)m$ is an integer, prove that the probability that RA satisfies at least $(1 - 1/k)m$ edges is at least $1/m$.

   **solution:**
   $pr(\text{edge i is satisfied}) = 1 - pr(\text{same color for u and v}) = 1 - 1/k$

   $E(\text{number of satisfied edges}) = \sum_{i=1}^{m}(1 - 1/k) = m(1 - 1/k)$

   $E(\text{number of satisfied edges}) = \sum j \times p_j = \sum_{j=1}^{m(1-1/k)-1} j \times p_j + \sum_{j=m(1-1/k)}^{m} j \times p_j =$

   $\leq (m(1 - 1/k) - 1) \sum_{j=1}^{m(1-1/k)-1} p_j + m \sum_{j=m(1-1/k)}^{m} p_j$

   $\leq (m(1 - 1/k) - 1) + m \times pr(\text{RA satisfies at least } (1 - 1/k)m \text{ edges})$

   $pr(\text{RA satisfies at least } (1 - 1/k)m \text{ edges}) \geq 1/m$

   (b) Give a Monte Carlo algorithm which satisfies at least $(1 - 1/k)m$ edges with probability at least $1 - 1/m$. Give a Las Vegas algorithm which satisfies at least $(1 - 1/k)m$ edges.

   **solution:**
   Mont Carlo Algorithm:
   from a, the probability that at least $(1 - 1/k)m$ edges are satisfied is at least 1/m for each run of RA. This probability that after q trails get $(1 - 1/k)m$ is $1 - (1 - 1/m)^q$. Find q so that:

   $1 - (1 - 1/m)^q \geq 1 - 1/m$

   $(1 - 1/m)^q \leq 1/m$

   $q \ln(1 - 1/m) \leq ln(1/m)$

   since $\ln(1 - 1/m) < 0$:

   $q \geq \frac{ln(1/m)}{1-1/m)}$

   if $q = 1 + \frac{ln(1/m)}{1-1/m)}$, the probability to get at least $(1 - 1/k)m$ is at least $1 - 1/m$

Las Vegas Algorithm:
Run the RA algorithm as many time as needed to have at least $(1 - 1/k)m$ edges satisfied. This probability is larger than zero so we will get there at some point.

8. 10 points

   Derandomize the randomized algorithm RA in the previous question to get an $O(km)$ time deterministic $(1 - 1/k)$-approximation algorithm for the maximization problem for $k$ colors and a graph $G$ of $n$ nodes and $m$ edges, and analyze the algorithm.

   **solution:**
   follow similar method to de-randomizing 3-CNF problem:

   Let $G = (V, E)$ be a graph with $|V| = n$ and $|E| = m$, and let $k > 1$ be the number of colors. Recall RA: each vertex is colored independently and uniformly at random from $\{1, \ldots, k\}$. For an edge $e = \{u, v\}$, the probability that $u$ and $v$ receive different colors is

$$\Pr[e \text{ is satisfied}] = 1 - \Pr[u \text{ and } v \text{ have the same color}] = 1 - \frac{1}{k}.$$

   Thus, if $X$ is the random variable "number of satisfied edges under RA",

$$E[X] = \sum_{e \in E} \Pr[e \text{ is satisfied}] = m\Big(1 - \frac{1}{k}\Big).$$

   We now derandomize RA using the method of conditional expectations.

   **Deterministic algorithm (RA′).**

   We color the vertices one by one in an arbitrary order. Suppose we have already assigned colors to a subset $S \subseteq V$ of vertices and let

$$\mu(S) := E[X \mid \text{colors on } S \text{ are fixed as chosen}].$$

   Initially, $S = \emptyset$ and $\mu(\emptyset) = E[X] = (1 - 1/k)m$.

   Now consider the next uncolored vertex $v \in V \setminus S$. For each color $c \in \{1, \ldots, k\}$, define

$$\mu_c := E[X \mid \text{colors on } S \text{ as before and } v \text{ is colored } c],$$

   where all remaining uncolored vertices are still colored at random as in RA.

   Since in RA the color of $v$ is chosen uniformly at random,

$$\mu(S) = E[\,\mu_c\,] = \frac{1}{k}\sum_{c=1}^{k} \mu_c.$$

   Therefore, there must exist a color $c^*$ with $\mu_{c^*} \geq \mu(S)$. Algorithm RA′ chooses such a color $c^*$ deterministically, sets $\mathrm{col}(v) = c^*$, and updates $S \leftarrow S \cup \{v\}$.

This guarantees that after each step the conditional expectation does not decrease:

$$\mu(S \cup \{v\}) = \mu_{c^*} \geq \mu(S).$$

After all $n$ vertices are colored, there is no randomness left, and $\mu(V)$ equals the actual number of satisfied edges of the final coloring, say $X^\star$. By monotonicity of the sequence $\mu(\emptyset) \leq \mu(S_1) \leq \cdots \leq \mu(V)$, we get

$$X^\star = \mu(V) \ \geq \ \mu(\emptyset) = E[X] = \Big(1 - \frac{1}{k}\Big)m.$$

Hence RA$'$ is a deterministic $(1 - 1/k)$-approximation algorithm.

**Running time.**

It remains to show that we can compute the values $\mu_c$ fast enough to obtain an $O(km)$-time algorithm.

Fix a step where some vertex $v$ is to be colored and a partial coloring on $S$ is given. Only edges incident to $v$ can change their contribution to the conditional expectation when we decide the color of $v$; all other edges keep the same conditional expected contribution.

Consider an incident edge $e = \{v, u\}$:

- If $u$ is uncolored, then regardless of the color assigned to $v$, $e$ has one colored and one random endpoint, so Pr[$e$ is satisfied] $= 1 - 1/k$; its contribution remains unchanged.
- If $u$ is already colored with color $d$:
  - Before coloring $v$, $e$ has one fixed and one random endpoint, so its expected contribution is $1 - 1/k$.
  - If we color $v$ with $c = d$, then $e$ is never satisfied, so its contribution becomes 0.
  - If we color $v$ with $c \neq d$, then $e$ is always satisfied, so its contribution becomes 1.

Thus, to compute all $\mu_c$ we need only look at neighbors of $v$ that are already colored. Let $d_v$ be the number of already-colored neighbors of $v$. We can scan all $d_v$ incident edges once, count for each color $c$ how many neighbors of $v$ currently use color $c$, and then compute all $\mu_c$ in $O(k)$ time from these counts.

Hence, the work for vertex $v$ is $O(d_v + k)$. Summed over all vertices,

$$\sum_{v \in V}(d_v + k) = \left(\sum_{v \in V} d_v\right) + kn = 2m + kn = O(km),$$

since $n \leq m$ for any connected graph with at least one edge, and in general $2m + kn = O(km)$.

Therefore RA$'$ runs in time $O(km)$ and outputs a coloring satisfying at least $(1-1/k)m$ edges, i.e., it is a deterministic $(1-1/k)$-approximation algorithm for the maximization problem.