Due 23:59 Dec 6 (Tuesday). There are 100 points in this assignment.
Submit your answers (**must be typed**) in pdf file to CourSys
`https://coursys.sfu.ca/2022fa-cmpt-705-x1/`. Submissions received after 23:59 of Dec 6 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 6, respectively; no points will be given to submissions after $00:30$ of Dec 6.

1. (Chapter 10 Problem 1 of the text book) 20 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.

   Answer We can prove that Hitting Set problem is NP complete by reducing another problem, Vertex Cover, to be solvable within polynomial time. We pose the problem as follows: a is an element in A ($a \in A$), which can be deleted from the set A and then deleting all sets from the other set B_i which contain a. This can easily be proven to be under polynomial time. We see that if we have a set B_i which is any of the sets in the Hitting sets ($B_i = \{x_1, x_2, ..., x_c\} \subseteq A$). With this given, we can safely say that we have at least one of these belong to the Hitting set. With this stated, we can say that B_i can be hit by k length H set if and only if for some value of i (iterator/subscript $i = 1, 2, ...c$) the instance reduced by x_i has a Hitting set H of length $k - 1$.

   We can model our algorithm to pick any set $B_i = \{x_1, x_2, ...x_i, ..., x_c\}$ and for each x_i we recursively check if the instance reduced by x_i has a Hitting set of $k - 1$ elements. If any of these recursive calls result in $True$, we return $True$. On worst case we will run the recursion a total of c times and each takes running complexity of $O(c)$ which makes it $O(c^{k+1})$, We need to do this a total of k times as we are checking if the set H is of size k or not. So the total complexity ends up being $O(k \times c^{k+1})$. This is the contribution of a function $f(c,k)$. The size of set A is n and there are a total of m collection of subsets of A, namely Bs. The function $p(n,m)$ is also a polynomial function. So to conclude, the running time is within the asked range.

   ---

2. (Chapter 10 Problem 3 of the text book) 15 points

   Give a digraph G of $V(G) = \{v_1, .., v_n\}$, we want to decide if G has a Hamiltonian path from v_1 to v_n. Give an $O(2^n p(n))$ time algorithm to solve the Hamiltonian path problem in G, where $p(n)$ is a polynomial in n.

Answer

In a Hamiltonian Path we have to remember the walk of the tree, but not the order of the walk. For this we have to construct a table (T) with indices $2^n \times n$. If we have a path in a subset of the graph G, lets call it $G[S]$. Say, we have two vertices $v_i \& v_j$ which are in this subset. The existence of paths in the set S can be denote by $T(S, v_i, v_j)$, essentially we are looking for the answer to $T(V, 1, n)$ (Path to all vertices). To determine if there is a path between any vertices $v_i \& v_j$, we have to look at all the nodes which have an edge (v_i, v_k) to another vertex v_k which then looks for a path to v_j which is only $True$ if there exists a path between v_k and v_j. This will take a running time of $O(n)$. To fill out the table T, we look for the smallest sets which have a Hamiltonian Path and then work our way up to bigger and bigger sets, till we reach to the answer to $T(V, 1, n)$. As we have n vertices, we will have to check for a path between v_i and v_j, n times. In the end we have to pass through the whole table T, which already takes $2^n \times n$ steps. The total running time for this algorithm is $O(2^n \times n^3)$, which is what was required.

We can pose this problem differently, where we find path which flows from vertices not more than once. The algorithm runs as follows:

Initialize value of T(v1, v) to 1 and the rest of them to be 0.
for i in 2 to n:
    for all S (subsets of V) such that $len(S) = i - 1$ for all $v_j \in V$
    if $T(S, v_j) = 1$
        for all $v_k \notin S$ such that (v_j, v_k) are edges
            $T(S \cup v_k, v_j) = 1$
if $T(n, v_i) = 1$ for $v_i \in V$
    return True
return False

---

3. (Chapter 10 Problem 5 of the text book) 15 points

   A dominating set D of a graph G is a subset of $V(G)$ such that for every node u of G, either $u \in D$ or u is adjacent to a node $v \in D$. The minimum dominating set problem in G is to find a minimum dominating set D of G (the problem is NP-complete). Give a polynomial time algorithm for the minimum dominating set problem in a tree.

   Answer

   ---

4. (15 points)

   Below is an algorithm to create a random permutation of n elements in an array $A[1..n]$.
   RANDOM-PERMUTE(A, n)
   for $i = 1$ **to** n
   select j uniformly at random from $\{i, i+1, .., n\}$ and exchange $A[i]$ with $A[j]$;

   Prove that the algorithm creates every permutation of $A[1..n]$ with probability$1/n!$.

   Answer

   ---

5. (20 points)

   Given a graph G, we consider the following problem: color the nodes of G using k 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) (14 points) 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. 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 $1/m$.

   (b) (6 points) Assume $(1 - 1/k)m$ is an integer. 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.

   Answer

   ---

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 $N(k)/k$ 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.

   (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.

   (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.

Answer
