Practice questions

Greedy algorithms

1. Prove by contradiction that for any cycle in a graph G whose edges have unique edge weights, the minimum spanning tree of G excludes the maximum-weight edge in that cycle
   Denote the maximum-weight edge in that cycle as $e = (u, v)$. For the sake of contradiction, assume that G has an MST T that contains e. Removing edge e from T will separate the tree into two parts. Because e is part of the cycle, there must be another path from u to v that does not involve e. This path will have one edge f that connects these two parts. Adding f back to T will connect the two disconnected part of T and create a cheaper spanning tree than T (because f is cheaper than e). This contradicts with T being an MST. As such, all minimum spanning tree of G must exclude e.

2. Under a Huffman encoding of n symbols with frequencies $f_1, f_2, ..., f_n$, what is the longest a codeword could possibly be? Give an example of frequencies that could produce this case.
   For a full binary tree of n leaves, the longest path will be of length $n - 1$. So the longest codeword we can have for a Huffman encoding of n symbols is of length $n - 1$. This can be achieved with the following frequencies: $1/2, 1/4, 1/8, ...1/2^{n-1}, 1/2^{n-1}$.

3. A server has n customers to serve. The required service time for each customer is known in advance: it is t_i minutes for customer i. So if, for example, the customers are served in increasing order of i, then the i-th customer will have to spend a total of $\sum_{k=1}^{i} t_i$ minutes to be completely done. We wish to minimize the total time spent by all the customers.

$$T = \sum_{i=1}^{n} (\text{time spent by customer } i)$$

   Give an efficient algorithm for computing the optimal order in which to process the customers.
   We simply proceed with a greedy strategy by sorting the customers in increasing order of their required service time and servicing them in this order. The running time of this will be $O(n \log n)$. To prove the correctness, for an arbitrary ordering of the customers, let $s(j)$ be the j-th customer in the ordering, we can rewrite the total wait time as follows:

$$T = \sum_{i=1}^{n} \sum_{j=1}^{i} t_{s(j)} = \sum_{i=1}^{n} (n - i - 1) t_{s(i)}$$

   For any ordering, if $t_{s(i)} \geq t_{s(j)}$ and $i \leq j$ (i.e., a customer of shorter service time is served after a customer of longer service time), we can always swap the position of the two customers and improve the total wait time. As such, the optimal order must satisfy $t_{s(1)} \leq t_{s(2)} \leq \cdots \leq t_{s(n)}$ and our greedy algorithm will give the correct optimal output.

Linear Programs

4. Duckwheat is produced in Kansas and Mexico and consumed in New York and California. Kansas produces 15 shnupells of duckwheat and Mexico 8. Meanwhile, New York consumes 10 shnupells and California 13. The transportation costs per shnupell are \$4 from Mexico to New York, \$1 from Mexico to California, \$2 from Kansas to New York, and \$3 and from Kansas to California. Write a linear program that decides the

amounts of duckwheat (in shnupells and fractions of a shnupell) to be transported from each producer to each consumer, so as to minimize the overall transportation cost.

Let $x_{i,j}$ be the number of shnupells of duckwheat produced in city i and consumed in city j.

$$\begin{array}{ll} \min & 4 \cdot x_{M,N} + 1 \cdot x_{M,C} + 2 \cdot x_{K,N} + 3x_{K,C} \\ s.t. & x_{K,N} + x_{K,C} \leq 15 \\ & x_{M,N} + x_{M,C} \leq 8 \\ & x_{K,N} + x_{M,N} \leq 10 \\ & x_{K,C} + x_{M,C} \leq 13 \\ & x_{K,N}, x_{M,N}, x_{M,N}, x_{M,C} \geq 0 \end{array}$$

5. A film producer is seeking actors and investors for his new movie. There are n available actors; actor i charges s_i dollars. For funding, there are m available investors. Investor j will provide p_j dollars, but only on the condition that certain actors $Lj \subseteq \{1, 2, ..., n\}$ are included in the cast (all of these actors L_j must be chosen in order to receive funding from investor j).

   The producer's profit is the sum of the payments from investors minus the payments to actors. The goal is to maximize this profit.

   Question: Express this problem as an integer linear program in which the variables take on values $\{0, 1\}$.

   Let $x_j = 1$ mean that investor j is investing in the movie, 0 otherwise. Let $y_i = 1$ mean that actor i is acting in the movie, 0 otherwise.

$$\begin{array}{ll} \max & \sum_{j=1}^{m} x_j p_j - \sum_{i=1}^{n} y_i s_i \\ s.t. & x_j |L_j| \leq \sum_{i \in L_j} y_i \\ & x_j, y_i \in \{0, 1\} \end{array}$$

6. What is the optimal solution to the following linear program:

$$\begin{array}{rl} \max & -x + y \\ s.t & 2x + 5y \geq 10 \\ & 2x - y \leq 20 \\ & y \leq 4 \end{array}$$

   A 5

   B 9 (Correct, achieved at $x = -5, y = 4$, which is obtained at the intersection of $y = 4$ and $2x+5y = 20$)

   C ∞

   D -8

   E 4

Computational complexity

7. Consider the following problem, which is known to be NP-complete:

   Hamiltonian Cycle
   Input: Undirected graph $G = (V, E)$
   Question: Does G has a cycle that visits each vertex in V exactly once?

   Consider Hamiltonian Cycle restricted to graphs in which every vertex has at most 2 edges. Call this problem Hamiltonian Cycle-2.

- Prove that Hamiltonian Cycle-2 is in NP.

  Given a proposed solution, which is a permutation of the nodes of the graph v_1, v_2, ..., v_n, we simply need to check if there exists an edge between v_i and v_{i+1} for $i = 1, ..., n-1$ and between v_n and v_1. If any edge is missing, it is an invalid solution, otherwise it is valid. This can clearly be done in linear time, showing that the problem is in NP.

- What is wrong with the following proof of NP-completeness for Hamiltonian Cycle-2?

  We know that the Hamiltonian Cycle problem is NP-complete, so it is enough to present a reduction from Hamiltonian Cycle-2 to Hamiltonian Cycle. Given a graph G with vertices of degree at most 2, the reduction simply leaves the graph unchanged to be the input for Hamiltonian Cycle. The answer to both problems would clearly be identical. This proves the correctness of the reduction, thus Hamiltonian Cycle-2 is NP-complete.

  The direction of the reduction is wrong. We need to reduce from HC to HC-2. To be complete, you should also show that HC-2 is in NP.

8. (a) I have a problem X that I can solve by using 3SAT as a black box, with an additional polynomial amount of time. (That is, problem X poly-time reduces to 3SAT.) Are the following statements true or false/unknown?

   i. X is in NP. *We can not draw any conclusion about whether X is in NP. So in this case, it is unknown.*

   ii. X is NP-hard. *We can not draw any conclusion about whether X is in NP-hard. We know that 3SAT is at least as hard as X but 3SAT could be a lot harder than X. Again it is unknown.*

   (b) I have two problems, X and Y. 3SAT reduces to problem X and problem X reduces to problem Y. (These reductions are poly-time reductions.) Is the following statement true or false/unknown?

   i. Y is NP-hard. *True. Because of the transitivity of reductions, we know 3SAT poly-time reduces to Y, so Y is at least as hard as 3SAT. Given that 3SAT is NP-hard, Y is also NP-hard.*

   (c) Problem X and problem Y are both NP-complete. Is the following statement true or false/unknown?

   i. Problem X reduces to problem Y and vice versa. *True. Because both problems are in NP, so Y being NP-complete suggests that X poly-time reduces to Y and vice versa.*

   For solutions, work through `https://courses.ecampus.oregonstate.edu/cs325/np-hard/story.html`

9. For each of the statements in this section, respond:
   A True B False C Unknown

   (a) Problem A poly-time reduces to problem B. If problem A is solvable in a polynomial time then problem B is solvable in polynomial time. *False. We can find counter examples to this statement. For example, k-clique, the problem of wether there is a clique of size k in a given graph, has a simple polynomial run time algorithm that simply checks all subgraph of size k in $O(n^k k^2)$ time. k-clique can be reduced to the general clique problem. But the general clique problem has not known polynomial run time algorithm. Now you can also say, but we don't know for sure that clique does not have poly-time algorithm. So if your answer is unknown, this is also acceptable.*

   (b) Problem A poly-time reduces to problem B. If problem A is NP-complete then problem B is NP-complete. *False. We can only conclude that B is NP-hard, but B may not be in NP.*

   (c) Problem A poly-time reduces to problem B. If problem A is NP-hard then problem B is NP-hard. *True*

   (d) If P $\neq$ NP, then the satisfiability problem (SAT) cannot be solved in polynomial time. *True*

   (e) P $\subseteq$ NP. *True*

   (f) NP-complete $\subseteq$ NP. *True*

   (g) The intersection of NP-complete and P is empty. *Unknown. Whether this is true will directly hinged upon the answer whether $P \neq NP$. Since we don't know that answer, we don't know if the statement is true either.*

(h) If the satisfiability problem (SAT) poly-time reduces to problem Y, then Y is NP-hard. *True*

(i) NP-complete is a subset of NP-hard. *True*

(j) If problem X is in NP and X poly-time reduces to the satisfiability problem (SAT), then X is NP-complete. *False. Wrong direction.*

(k) No problems in NP can be solved in polynomial time. *False. Note that $P \subseteq NP$. A good portion of problems in NP are solvable in poly-time.*

(l) There are problems in NP that cannot be solved in polynomial time. *Don't know. Depends on wether $P \neq NP$.*

(m) If the satisfiability problem (SAT) can be solved in polynomial time, then all problems in NP can be solved in polynomial time. *True.*

For the last two questions, consider the following procedure:

```
procedure(X)
    Y[0] = X
    for i = 1 to k
        Y[i] = subroutine(Y[i-1])
```

Assume that `X` is valid input to `procedure`, `Y[i-1]` is valid input to `subroutine` for every `i`, and `subroutine` is a polynomial time algorithm.

(n) If `k` is a constant, then `procedure` is a polynomial time algorithm. *True. Constant number of calls of a poly-time algorithm still runs in poly-time.*

(o) If `k` is a polynomial function of the size of `X`, then `procedure` is a polynomial time algorithm. *True. Polynomial number of calls of a poly-time algorithm still runs in poly-time.*

10. We are feeling experimental and want to create a new dish. There are various ingredients we can choose from and we'd like to use as many of them as possible, but some ingredients don't go well with others. If there are n possible ingredients (numbered 1 to n), we write down an $n \times n$ binary matrix where the (i, j) entry is 1 if i and j can go together and 0 otherwise. Notice that this matrix is necessarily symmetric; and that the diagonal entries are always 0.

    We wish to solve the following problem:

    EXPERIMENTAL CUISINE :

    input: n, the number of ingredients to choose from; B, the $n \times n$ binary matrix that encodes which items go well together; and k the target number of ingredients.

    output: Can we come up with a dish with $\geq k$ ingredients that all go well with one another?

    Show that if EXPERIMENTAL CUISINE is NP-complete.

    We first show that EXPERIMENTAL CUISINE is in NP. A solution to an instance of the EPERIMENTAL CUISINE problem is a set of ingredients. To verify it, we just need to go through the conflict matrix to check if any two ingredients in the list are in conflict with each other. This verification can be complete in $O(n^2)$ time.

    To show that EXPERIMENTAL CUISINE is NP-hard, we will use reduction from INDEPENDENT SET: we can use an algorithm that solves EXPERIMENTAL CUISINE to solve INDEPENDENT SET.

    For INDEPENDENT SET, we wish to know if a graph G has a set of k vertices such that no two vertices are adjacent. To solve this, we create an instance of the EXPERIMENTAL CUISINE problem. For each vertex of the graph, we create an ingredient. The matrix B is the complement of the adjacency matrix for G: $B(i, j)$ is 0 if there is an edge between vertices i and j in G (in this case we would not be allowed to select these ingredients). It is evident that the group G will have an independent set of size k or larger if and only if in the constructed EXPERIMENTAL CUISINE problem , we can create a dish with k ingredients or more. And the reduction is done in $O(n^2)$ time where n is the number of nodes in the graph.

    Thus it follows that EXPERIMENTAL CUISINE is NP-complete.