Homework 4

Jonathan Henke, Tadeo Zuniga, Addison Marcus

April 2023

Question 1:

Band Tour Problem. A band (called "FPT") is planning to go on tour and they have a list of cities that they would like to play in. They are on a tight budget, so they don't want to spend more than a few hundred dollars to fly to the next city. Suppose that they want to visit n cities, and suppose that they have a list of all city pairs for which it is possible to get cheap flights. The band would like to know whether it is possible to visit each city exactly once by only taking cheap flights. Assume they don't care which city they start in. They would like to solve their problem in O(2^n poly(n)) time (thus living up to their name). Can you provide an algorithm for them to use?

Algorithm 1 BAND TOUR PROBLEM

```
Require: A set of $n$ cities $C$ and a list of cheap flights between pairs of cities $E$
Ensure: A path that visits each city in $C$ exactly once using only cheap flights, or report that no such path exists
  Let $H(S, v)$ be true if and only if there is a path that visits all cities in $S$ exactly once and ends at $v$
  Initialize $H(v, v) \leftarrow$ true for all cities $v \in C$
  for $k = 2$ to $n$ do
    for each subset $S \subseteq C$ of size $k$ do
      for each $v \in S$ do
        $H(S, v) \leftarrow$ false
        for each $u \in S$, $u \neq v$ do
          if there exists an edge $(u, v) \in E$ and $H(S \setminus v, u)$ is true then
            $H(S, v) \leftarrow$ true
            break
          end if
        end for
      end for
    end for
  end for
  for each $v \in C$ do
    if $H(S, v)$ is true then
      return "a path exists to all cities"
    end if
  end for
  return "no such path exists"
```

The algorithm employs dynamic programming to solve the problem of finding a path that visits each city exactly once. It uses a boolean table $H(S, v)$ to indicate whether a path exists that visits all cities in subset S and ends at city v. The algorithm iteratively computes the values of $H(S, v)$ for all subsets S of size k and cities v in S.

The algorithm has an outer loop that ranges from 2 to n, the number of cities. The second loop iterates over all subsets S of size k, and the innermost loop iterates over all cities v in subset S. The total number of iterations of the innermost loop is $O(n^2)$. The total runtime of the algorithm is $O(2^n n^2)$, as the number of iterations of the innermost loop is exponential in the number of cities.

Question 2:

Art Gallery Problem revisited. Suppose you are still in the art gallery security consulting business but now instead of human guards (that could only watch a single art piece), you can install special video cameras that can monitor multiple works of art at the same time (provided the camera can see them). Your business would like to come up with a quick (polynomial-time) estimate of the number of cameras needed in order to guarantee every work of art is monitored. How would you propose to solve the problem?

A linear program can be used to solve this problem. We are wanting to find the smallest set of cameras so every artwork in the gallery is being watched. This can be expressed as an integer linear program, which then could be relaxed to a continuous constraint. The resulting linear program could then be solved. The runtime of the linear program should be should be polynomial if solved properly.

Let C be the set of all possible camera positions, A be the set of all art in the gallery, and the cost of each camera would be 1. We want to find the smallest set of cameras such that every artwork in A is visible by at least one camera in C. For each artwork j, let $N(j)$ be the set of cameras that can see artwork j. Let x_i be an indicator variable that lets us know if a camera is chosen or not. Then we can write the following integer linear program:

Minimize: $\sum_{i \in C} x_i$
Subject to: $\sum_{i \in N(j)} x_i \geq 1 \quad \forall j \in A$
$x_i \in 0, 1 \quad \forall i \in C$

Question 3:

Hitting Set as ILP. The hitting set problem is defined as follows: you have a set of elements $X = \{x_1, \dots, x_n\}$ and collection of sets $S = \{S1, \dots, S_m\}$. The goal is to find a subset H of X, such that H has a non-empty intersection with each S_j in S and —H— is minimized.

(a). Express Hitting Set as an Integer Linear Program (ILP).

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} x_i && && (1) \\ \text{subject to:} \quad & \sum_{i:x_i \in S_j} x_i \geq 1, && \forall_j \in \{1, \dots, m\} && (2) \\ & x_i \in \{0, 1\}, && \forall_i \in \{1, \dots, n\} \end{aligned}$$

In summary, the Hitting Set problem can be expressed as an ILP by defining binary variables x_i for each element in the set X, where x_i equals 1 if x_i is in the hitting set H, and 0 otherwise. The objective function of the ILP minimizes the size of the hitting set H, which is equal to the sum of all x_i. To ensure that each set S_j has at least one element in H, the ILP includes a constraint that requires the sum of all x_i that are in S_j to be greater than or equal to 1. Additionally, the variables are restricted to binary values. By solving this ILP, we can find the optimal hitting set for the given instance of the problem.

(b). Suppose each S_j contains at most M elements. Convert the ILP above to a 'relaxed' LP, then devise a rounding scheme to round the solution back to an integer solution. Argue that this provides an M-approximation algorithm for Hitting Set.

Suppose each S_j contains at most M elements. We can convert the ILP formulation of Hitting Set into a relaxed LP by relaxing the binary constraints $x_i \in 0, 1$ to $0 \leq x_i \leq 1$. The resulting LP is:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} x_i \\ \text{subject to:} \quad & \sum_{i:x_i \in S_j} x_i \geq 1, && \forall_j \in 1, \dots, m \\ & 0 \leq x_i \leq 1, && \forall_i \in 1, \dots, n \end{aligned}$$

We can then use a rounding scheme to obtain an integer solution. One such scheme is if we know that each set S_j contains at most M elements, we can set a threshold of $\frac{1}{M}$. That is, we set x_i to 1 if $x_i \geq \frac{1}{M}$, and to 0 otherwise.

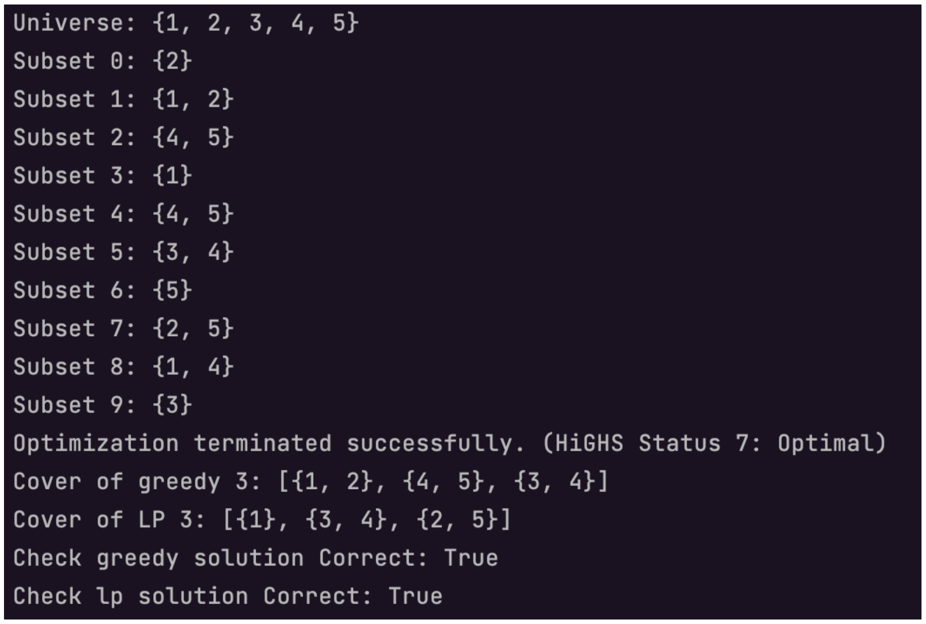
By using this rounding scheme, we can show that the size of the resulting hitting set is at most M times the size of the optimal fractional solution. This means that the rounding scheme provides an upper bound on the size of the hitting set that is at most M times the size of the optimal hitting set. Since we want to minimize the size of the hitting set, this means that the rounding scheme gives us a solution that is at most M times larger than the optimal solution. Therefore, we call it an M-approximation algorithm.

Programming:

Programming project (15 pts):

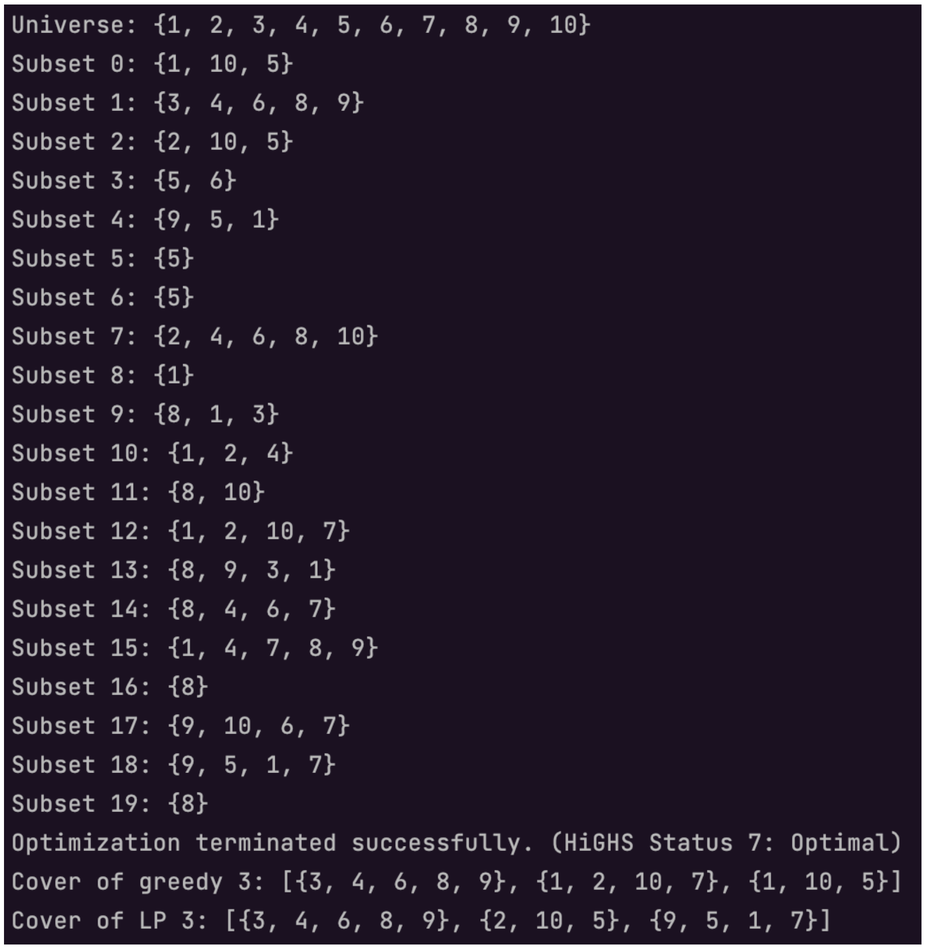
For the greedy algorithm, we used a set data structure and just added the set that had the most elements that weren't in a set called covered. This was done by removing elements in the covered set from each set and getting the lengths of each set without those elements and then adding the elements in the subset that were not already in covered to covered.

Covered Set for Universe of 5 with 10 subsets



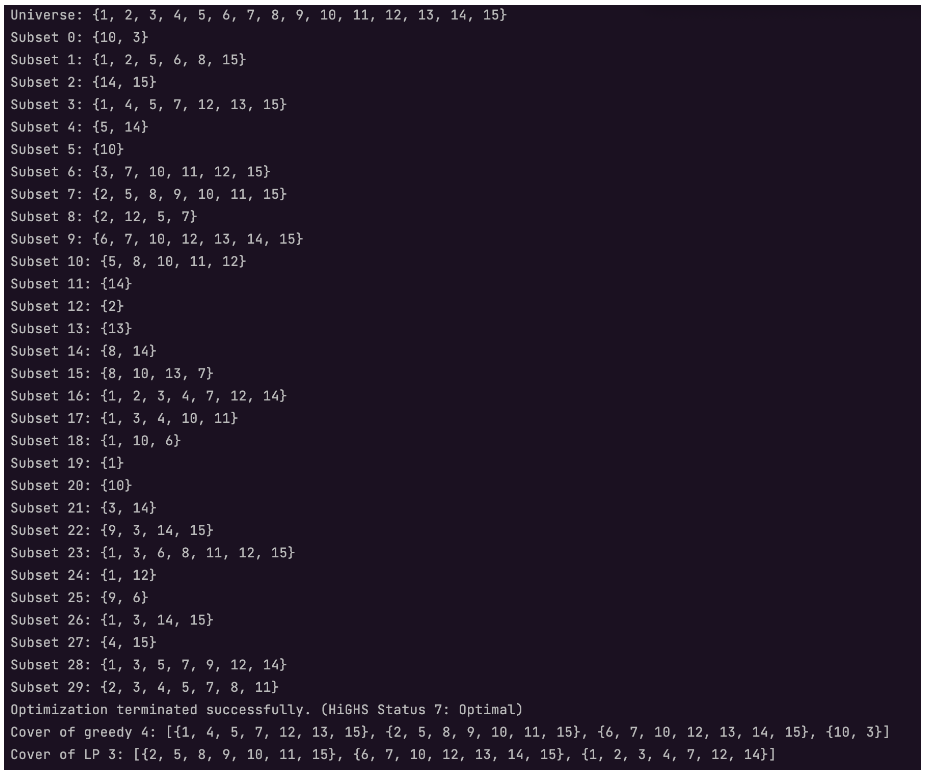
```
Universe: {1, 2, 3, 4, 5}
Subset 0: {2}
Subset 1: {1, 2}
Subset 2: {4, 5}
Subset 3: {1}
Subset 4: {4, 5}
Subset 5: {3, 4}
Subset 6: {5}
Subset 7: {2, 5}
Subset 8: {1, 4}
Subset 9: {3}
Optimization terminated successfully. (HiGHS Status 7: Optimal)
Cover of greedy 3: [{1, 2}, {4, 5}, {3, 4}]
Cover of LP 3: [{1}, {3, 4}, {2, 5}]
Check greedy solution Correct: True
Check lp solution Correct: True
```

Covered Set for Universe of 10 with 20 subsets



```
Universe: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
Subset 0: {1, 10, 5}
Subset 1: {3, 4, 6, 8, 9}
Subset 2: {2, 10, 5}
Subset 3: {5, 6}
Subset 4: {9, 5, 1}
Subset 5: {5}
Subset 6: {5}
Subset 7: {2, 4, 6, 8, 10}
Subset 8: {1}
Subset 9: {8, 1, 3}
Subset 10: {1, 2, 4}
Subset 11: {8, 10}
Subset 12: {1, 2, 10, 7}
Subset 13: {8, 9, 3, 1}
Subset 14: {8, 4, 6, 7}
Subset 15: {1, 4, 7, 8, 9}
Subset 16: {8}
Subset 17: {9, 10, 6, 7}
Subset 18: {9, 5, 1, 7}
Subset 19: {8}
Optimization terminated successfully. (HiGHS Status 7: Optimal)
Cover of greedy 3: [{3, 4, 6, 8, 9}, {1, 2, 10, 7}, {1, 10, 5}]
Cover of LP 3: [{3, 4, 6, 8, 9}, {2, 10, 5}, {9, 5, 1, 7}]
```

Covered Set for Universe of 15 with 30 subsets



```
Universe: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
Subset 0: {10, 3}
Subset 1: {1, 2, 5, 6, 8, 15}
Subset 2: {14, 15}
Subset 3: {1, 4, 5, 7, 12, 13, 15}
Subset 4: {5, 14}
Subset 5: {10}
Subset 6: {3, 7, 10, 11, 12, 15}
Subset 7: {2, 5, 8, 9, 10, 11, 15}
Subset 8: {2, 12, 5, 7}
Subset 9: {6, 7, 10, 12, 13, 14, 15}
Subset 10: {5, 8, 10, 11, 12}
Subset 11: {14}
Subset 12: {2}
Subset 13: {13}
Subset 14: {8, 14}
Subset 15: {8, 10, 13, 7}
Subset 16: {1, 2, 3, 4, 7, 12, 14}
Subset 17: {1, 3, 4, 10, 11}
Subset 18: {1, 10, 6}
Subset 19: {1}
Subset 20: {10}
Subset 21: {3, 14}
Subset 22: {9, 3, 14, 15}
Subset 23: {1, 3, 6, 8, 11, 12, 15}
Subset 24: {1, 12}
Subset 25: {9, 6}
Subset 26: {1, 3, 14, 15}
Subset 27: {4, 15}
Subset 28: {1, 3, 5, 7, 9, 12, 14}
Subset 29: {2, 3, 4, 5, 7, 8, 11}
Optimization terminated successfully. (HiGHS Status 7: Optimal)
Cover of greedy 4: [{1, 4, 5, 7, 12, 13, 15}, {2, 5, 8, 9, 10, 11, 15}, {6, 7, 10, 12, 13, 14, 15}, {10, 3}]
Cover of LP 3: [{2, 5, 8, 9, 10, 11, 15}, {6, 7, 10, 12, 13, 14, 15}, {1, 2, 3, 4, 7, 12, 14}]
```