CS218 Entrance Report

due Sunday, April 7, 23:59 PM

Problem A:
To be print in the correct format, each student is placed in its respective house using a python dictionary. Then, each list in the dictionary is sorted in alphabetical order. Finally, the house and students are printed in the format specified by the problem description. Runtime: $O(n * log(n))$ due to the sorting algorithm from Python; Space Complexity: $O(n)$, the total number of students.

Problem B:
To determine the maximum number of items Harry can buy, it is the same as asking how many of the cheapest items can Harry buy. Sort the prices in increasing order. Then, "purchase" these items in order until the running total is larger than k, the amount of money Harry has. Runtime: $O(n * log(n))$ due to the sorting algorithm from Python; Space Complexity: $O(n)$, the total number of items.

Problem C:
Dynamic programming is used for the knapsack problem. Each cell of the table represents the maximum value with a subset of items and weight limit. If the weight of the current cell exceeds the capacity w, the algorithm "chooses" not to include that item by copying over the $[i-1][w]$ cell. Otherwise, a comparison is done between the value from including OR not including the item, and the max is chosen. The bottom right cell $[n][W]$ contains the max value obtained using all n items with a W weight limit. Runtime: $O(nW)$ due to filling the dynamic programming table; Space Complexity: $O(nW)$, the size of the dynamic programming table.

Problem D:
The problem is split into two: longest increasing and longest decreasing subsequences. Then the maximum length is taken from the increasing and decreasing arrays, and minus 1 to prevent double counting. Runtime: $O(n^2)$ due a nested for loop; Space Complexity: $O(n)$, both array are of size n.

Problem E:
A* search is used to solve the trip navigator problem. A priority queue is used to explore the grid using a Manhattan distance heuristic. There is a set visited to track all visited cells and dictionary shortest_paths to memoize the shortest paths. Runtime depends on the heuristic's effectiveness; Space Complexity: $O(n^2)$ as that is the size of the grid.