2. Describe an efficient greedy algorithm that, given a set $\{x_1, x_2, \ldots, x_n\}$ of points on the real line, determine the smallest set of unit-length closed intervals that contains all of the given points. Prove that your answer is correct.

Fewest Intervals Algorithm:

Input: Set $\{x_1, x_2, \ldots, x_n\}$ of points

Output: Set of unit-length closed intervals that contains all the given points

Algorithm:

```
    Let P = $\{x_1, x_2, \ldots, x_n\}$ be the input set of points
    Let I be an initially empty set of intervals
    Sort P to be in ascending order
    While P is not empty:
        Take $x_0$ to be the smallest element in P
        Create interval [$x_0$, $x_0$+1] and add it to I
        Remove all points that appear in [$x_0$, $x_0$+1] from P
    Return I
```

Proof of correctness:

Let x_0 be the leftmost point of the input set. Suppose S is an optimal set of intervals such that x_0 is contained in the interval [a, a+1], where a<x_0. Since x_0 is the leftmost point of the input set, any points in [a,a+1] will also be in [x_0, x_0+1]. Therefore, [x_0, x_0+1] must still be optimal since it contains all of the points from the interval [a,a+1] found in our optimal solution. Since we have an optimal interval for [x_0, x_0+1], we can consider the next subproblem of the optimal interval for x_1. Since our choice for the interval of x_0 leads to the global optimal solution and since our optimal solution set S contains the optimal solutions to all subproblems, this problem has the greedy choice property and optimal substructure. Our algorithm finding the optimal solution for each subproblem along with the greedy choice property and optimal substructure of the problem implies that our algorithm is correct.

3. For the knapsack problem, given a set of items with the following weight and value combos and the weight capacity of the backpack as 100, provide the optimal solution for four cases: fractional knapsack with repetitions, fractional knapsack without repetitions, 0/1 knapsack with

repetitions, 0/1 knapsack without repetitions. The list of items in form {weight, value} is: {5,30}, {6,35}, {7,32}, {8,38}, {9,40}, {10,41}, {11,45}, {12,46}, {13,50}, {15,55}, {16,60}, {20,80}.

3.1. Fractional knapsack with repetitions:

To solve this, we will just take as many copies of the "most efficient" item as we need to fill our 100 weight knapsack. The most efficient item is the one with the highest value per unit weight. We will construct a table to find this.

{5, 30} = 6 val/weight

{6, 35} ≈ 5.83 val/weight

{7, 32} ≈ 4.57 val/weight

{8, 38} = 4.75 val/weight

{9, 40} ≈ 4.44 val/weight

{10, 41} = 4.1 val/weight

{11, 45} ≈ 4.09 val/weight

{12, 46} ≈ 3.83 val/weight

{13, 50} ≈ 3.85 val/weight

{15, 55} ≈ 3.67 val/weight

{16, 60} = 3.75 val/weight

{20, 80} = 4 val/weight

Since {5,30} has the highest value per weight, we will fill our backpack with copies of it. 100/5 = 20, so we can fit 20 copies of {5,30} in our backpack for a total value of 20*30 = 600.

3.2 Fractional knapsack without repetitions

We simply add the items in the order of decreasing val/weight. If the full item doesn't fit, we will add a fractional amount of it. We add: {5, 30}, {6,35}, {8, 38}, {7, 32}, {9, 40}, {10, 41}, {11, 45}, {20, 80}, {13, 50}. We have 11 weight left and the next item is {12, 46}, so we will take 11/12 of it resulting in {11, 253/6} being added. The value of our result is 2599/6 ≈ 433.1667.

3.3 0/1 Knapsack with repetitions

This is easy because our highest val/weight item is {5, 30} and since 5 divides 100 evenly, we take 20 copies of {5, 30} and get value 600, the same as in part 1.

3.4 0/1 Knapsack without repetitions

This problem is much harder and is best to do with a program. By using the “Knapsack.java” code found with this submission, we obtain a value of 426 with items: {60, 16}, {55, 15}, {50, 13}, {45, 11}, {41, 10}, {40, 9}, {38, 8}, {32, 7}, {35, 6}, {30, 5}.