Evan Zhang | z5383657

Question 1

1.1

At the first vase, put 1 flower in, then 2 at the next vase, then 3 at the one after, etc. until all vases have been visited and/or all flowers are gone.

If flowers run out before all vases have their assigned number of flowers put in, then no such assignment exists. This is because this is the minimum amount of flowers for each vase to contain a different number of flowers but each vase must have at least 1 flower, which the remaining vases have 0.

If there are still flowers at the end, put the rest in the last vase. This satisfies all requirements as:

1. All flowers are placed in a vase
2. The remaining flowers cannot be greater than n (i.e. the starting number of flowers) and therefore can be put into the last vase
3. Since we are increasing the flowers for each vase, there is no vase prior with the same number of flowers.

Since each vase (there are k vases) is visited once and flower number per vase is computed in O(1) speed, the above algorithm runs in O(k) speed.

1.2

Add all f_i together, taking O(k) time.

If the result is less than n flowers, no such assignment exists.

If the result is equal to n flowers, assign each vase it's maximum capacity. Sort the f_i array in O(klogk) time and scan through for repeats. If there are no repeats, the goal has been achieved, otherwise, no such assignment exists as the third criteria is not met.

If the result is more than n flowers, proceed to below.

Pair the vases with their f_i and sort in ascending order of f_i size, taking O(klogk) time.

Repeat the algorithm described in 1.1. If at any point does the preassigned number of flowers exceed the vase's maximum capacity, no such assignment exists. This is because the vases are sorted by capacity and the algorithm in 1.2 places the minimum amount of flowers needed to satisfy the requirements.

At the end, if there are flowers left and the last vase cannot hold them all, put in the maximum amount of flowers starting from the last vase going back to the first vase. If the maximum is equal to the previous maximum, put in 1 less flower than the current maximum. Repeat this if there are multiple maximums that are equal in a row. If the last vase is filled based on the previous criteria and there are flowers still available, no such assignment is possible. This is because the maximum amount of flowers have been put in vases but flowers still remain un-vased.

If the flowers run out before the first vase is filled, the goal has been completed. This is because:

1. All flowers are put in a vase

2. All vases have at least 1 and less than or equal to f_i flowers
3. No vases will have the same number of flowers

This algorithm takes O(klogk) time to sort and O(2k) time to put flowers in each vase. Therefore the algorithm takes O(klogk) time.