ComS 311
PA1
Adam Kiel
Zachary Schreck

WarWithArray:

- Compute2k pseudo code
  - Loop over elements of S, creating every possible combination by using another for loop to loop over every other element of S
  - For each combination, traverse over the String, checking every substring of length K to see if it is in S, if it is then add it to the arrayList T
  - Return arrayList T
- Runtime of compute2k
  - Generating each possible combination is O(n^2)
  - Verifying every substring in each combination is O(kn)
  - Since we verify every substring while generating each possible combination, our total runtime is O(k*n^3)

WarWithBST:

- Compute2k pseudo code
  - Loop over elements of S, creating every possible combination by using another for loop to loop over every other element of S
  - For each combination, traverse over the String, checking every substring of length K to see if it is in S by searching for it in the BST, if it is, then add it to the arrayList T
  - Return arrayList T
- Runtime of compute2k
  - Generating each possible combination is O(n^2)
  - Verifying every substring in each combination is O(kn) [being expressed as n here, but it is actually k* h where h is height of bst, but since worst case would be h = n, n is our representation]
  - Since we verify every substring while generating each possible combination, our total runtime is O(k*n^3)

WarWithHash:

- Compute2k pseudo code
    - Loop over elements of S, creating every possible combination by using another for loop to loop over every other element of S
    - For each combination, traverse over the String, checking every substring of length K to see if it is in S by searching for it in the HashSet, if it is then add it to ArrayList T
    - Return ArrayList T
- Runtime of compute2k
    - Generating each possible combination is O(n^2)
    - Verifying every substring in each combination is O(k^2) [assuming HashSet search is O(k), where k is time to compute the hash]
    - Since we verify every substring while generating each possible combination, our total runtime is O((k^2)*(n^2))

WarWithRollHash:

- Compute2k pseudo code
    - Loop over elements of S, creating every possible combination by using another for loop to loop over every other element of S
    - For each combination, use a rolling hash to check every substring, and see if that is in our HashSet, this takes O(k) for the time to roll over the entire string. Searching in the HashSet now takes O(1) instead of O(k) since we are passing in the hashed key instead of having to compute it on the fly
    - Since we verify every substring while generating each possible combination, our total runtime is O(k*n^2)