WarWithArray.

PseudoCode:

```
Input: int k, Array a[]\\
For each string e in a[]:
    For each string f in a[]:
        check if the string "e+f" is valid(e+f)


function valid(s):
   For each b = (k length substring in s):
       Loop through our original array a:
           check if b is in a
```

Runtime of compute2k():
The algorithm essentially 4 nested loops, they run n-times, n-times, k-times, and n-times.
So the run-time is $O(kn^3)$

WarWithBST.

PseudoCode:

```
Input: int k, Array a[]\\
For each string e in a[]:
    For each string f in a[]:
        check if the string "e+f" is valid(e+f)


function valid(s):
   For each b = (k length substring in s):
       search the BST for the substring b
```

Runtime of compute2k():
The algorithm essentially 4 nested loops, they run n-times, n-times, k-times, and log n-times.
So the run-time is $O(kn^2 logn)$

WarWithHash.

PseudoCode:

```
Input: int k, Array a[]\\
For each string e in a[]:
    For each string f in a[]:
        check if the string "e+f" is valid(e+f)


function valid(s):
   For each b = (k length substring in s):
       hash b and look it up in the HashSet
```

Runtime of compute2k():
The algorithm essentially 4 nested loops, they run n-times for the most outer loop, n-times for the second most outer loop, k-times for each substring of the 2k string, and k-times for the hash of the k-length string. So the run-time is $O(k^2n^2)$

WarWithRollHash.

PseudoCode:
Runtime of compute2k():