Pseudo-Code

**WarWithArray**

// Instance variables and constructor

public ArrayList<String> compute2k()

{

loop i = 0 through S.length

loop j = i through S.length

if(valid())

add 2k length string to T

return T;

}

private valid(String a, String b)

{

Collect all substrings shared between a and b

Loop

Loop

if there are equal number of valid strings in S as there are strings

shared between a and b, then they are a valid 2k substrings // return true

}

**WarWithBST**

Construct tree from input string

public ArrayList<String> compute2k()

{

loop i = 0 through S.length

loop j = i through S.length

if(valid())

add 2k length string to T

return T;

}

private valid(String a, String b)

{

Collect all substrings shared between a and b

Loop

Use BST search method, if BST search finds all the substrings shared between a and b

Then they are a valid 2k substring

}

**WarWithHash**

Construct hashtable from input string

public ArrayList<String> compute2k()

{

loop i = 0 through S.length

loop j = i through S.length

if(valid())

add 2k length string to T

return T;

}

private valid(String a, String b)

{

Collect all substrings shared between a and b

Loop

Use hash containsKey method, if table finds all the substrings shared between a and b

Then they are a valid 2k substring

}

**WarWithRollHash**

Construct hashtable from input string

public ArrayList<String> compute2k()

{

loop i = 0 through S.length

loop j = i through S.length

if(valid())

add 2k length string to T

return T;

}

private valid(String a, String b)

{

Collect all substrings shared between a and b

Loop

}

private long[] hashRolls()

{

Constructs an array which has the interim hash values between each set of keys

Allows for easy transition between keys while following Robin Karp behavior

}

**Asymptotic Run times:**

* WarWithArray
  + - Operation to go through and select strings to validate
    - Verifies whether substrings are valid
  + Overall : ***O(k)***
* WarWithBST
  + - Operation to go through and select strings to validate
    - Verifies whether substrings are valid
  + Overall : ***O(***
* WarWithHash
  + - Operation to go through and select strings to validate
    - Verifies whether substrings are valid
  + Overall : ***O(***
* WarWithRollHash
  + - Operation to go through and select strings to validate
    - Verifies whether substrings are valid
  + Overall

In the end the WarWithBST and WarWithHash performed best under these conditions. While both WarWithRollHash and WarWithArray required much more thorough traversals through their respective sets, the BST and Hashtable were able to more efficiently navigate to their targets through systematic avoidance and taking advantage of their logarithmic/constant time attributes.