Quiz 6

學號: 111550129　姓名: 林彥亨

Problem 1

The Walsh-Hadamard Transform (WHT) is a series of procedures that effectively calculates the Hadamard transform of a one-dimensional signal with real values.

To aid in comprehension, we present the pseudocode below for a simple implementation of the Discrete Walsh-Hadamard Transform.

Walsh-Hadamard Transform:

```
def WHT(x):
    # Function computes (slow) Discrete Walsh-Hadamard Transform
    # for any 1D real-valued signal
    # (c) 2015 QuantAtRisk.com, by Pawel Lachowicz
    x = np.array(x)
    if (len(x.shape) < 2):  # make sure x is 1D array
        if (len(x) > 3):  # accept x of min length of 4 elements (M=2)
            # check length of signal, adjust to 2**m
            n = len(x)
            M = math.trunc(math.log(n, 2))
            x = x[0:2 ** M]
            h2 = np.array([[1, 1], [1, -1]])
            for i in range(M - 1):
                if (i == 0):
                    H = np.kron(h2, h2)
                else:
                    H = np.kron(H, h2)

            return (np.dot(H, x) / 2. ** M, x, M)
```

a) Please showcase the **recursive process** of the Walsh-Hadamard Transform using the pseudocode provided above.

b) Examine different **applications** of the Walsh-Hadamard Transform, highlighting how its properties offer advantages in each specific application.

(a)

The recursive process of the Walsh-Hadamard Transform:

We supposed that the original signal has 8 elements. The base case starts with `h2`, and the recursive process would build `H` as follows:

1. The base case of the recursion is represented by the matrix `h2`, which is the smallest (2×2) Hadamard matrix.
2. The recursion occurs in the loop `for i in range(M - 1):`. In each iteration, the Hadamard matrix `H` is expanded by using the Kronecker product, denoted by `np.kron`, which is a tensor product for matrices. This operation combines two matrices to form a new, larger matrix.
3. On the first iteration (`i == 0`), `H` is set to the Kronecker product of `h2` with itself, effectively doubling its size. In each subsequent iteration, `H` is again expanded by the Kronecker product with `h2`. This is the recursive step, where the matrix `H` is built up from the base case `h2`.
4. After building the Hadamard matrix `H` to the appropriate size, the transform is completed by multiplying `H` with the input signal `x` and normalizing the result.

Here is the example:

1st recursion:

`H = kron(h2, h2) = [[1, 1, 1, 1], [1, -1, 1, -1], [1, 1, -1, -1], [1, -1, -1, 1]]`

2nd recursion:

`H = kron(H, h2)` — which would be an 8×8 matrix.

After the final recursion, the Hadamard matrix `H` would be an 8×8 matrix, which is then used to transform the input signal `x` by multiplying it with `H` and normalizing the result.

(b)

Some applications of the Walsh-Hadamard Transform:

- Image Processing → The Walsh-Hadamard Transform can provide fast and simple colculation to deal with images.
- Signal processing → The Walsh-Hadamard Transform can provide orthogonality and construction of Hadamard code to examine the error and correct the code.
- Data Compression → The Walsh-Hadamard Transform can provide the ability to compress data.
- Feature extraction → The Walsh-Hadamard Transform is able to address the binary pattern recognition.

- Quantum Computing → In quantum algorithms, based on the Walsh-Hadamard Tranform, we get the Hadamard gate that we used to create the superpositions of states.

Problem 2

The Miller-Rabin test is an algorithm based on probability that is employed to ascertain the primality of a given number. It operates by selecting random bases multiple times and examining if these bases offer substantial indications that the number is not prime.

The procedure consists of the following steps:

1. Given a number n, find an integer s and an odd number q such that $n-1=2^s q$.
2. Choose a random number a from the range $[1, n-1]$.
3. Compute $a^q \mod n$. If the result is 1 or $n-1$, then n passes.
4. For i from 0 to $s-1$, compute $a^{2^i q} \mod n$. If one of these is $n-1$, n is again passes.
5. If none of the above conditions are met, n is composite.

It is typical to carry out trial divisions by small primes before conducting the Miller-Rabin test in order to promptly identify obvious composites.

a) What **happens** when we apply the Miller-Rabin test to numbers in the format pq, where p and q are large prime numbers?

b) Can we **break** RSA with it?

(a)

Using the Miller-Rabin test to check a number *n* that is a product of two large primes *p* and *q*, the test will almost certainly determine that *n* is composite, which is the correct result. This is because the conditions for a number passing the Miller-Rabin test as potentially prime become increasingly unlikely as the number's factors grow larger.

In practice, through step 3 and step 4, the Miller-Rabin test is run multiple times with different random values of *a* to reduce the probability of a false positive (declaring a composite number as probably prime). The larger the number of tests, the higher the confidence in the result.

(b)

No, the Miller-Rabin test cannot be used to break RSA encryption. Because the Miller-Rabin test can only tell you that whether the number is composite with

high probability, in other words, whether a given number is prime or not. RSA security relies on the difficulty of factoring large numbers that are the product of two large prime numbers, all you need to break it is actually to find out the prime factors of the public key, not just determine that the number is composite or not.