Parallel FFT

Team Members:

Mourya Reddy - 201501240

Krishna Chaitanya Pappu - 201501236

Vadlamudi Avinash - 201501164

Algorithm:

Recursive Implementation:

```
X0,...,N-1 ← ditfft2(x, N, s):             DFT of (x0, xs, x2s, ..., x(N-1)s):
    if N = 1 then
        X0 ← x0                                      trivial size-1 DFT base case
    else
        X0,...,N/2-1 ← ditfft2(x, N/2, 2s)            DFT of (x0, x2s, x4s, ...)
        XN/2,...,N-1 ← ditfft2(x+s, N/2, 2s)          DFT of (xs, xs+2s, xs+4s, ...)
        for k = 0 to N/2-1                           combine DFTs of two halves into full DFT:
            t ← Xk
            Xk ← t + exp(-2πi k/N) Xk+N/2
            Xk+N/2 ← t - exp(-2πi k/N) Xk+N/2
        endfor
    endif
```

In the above method, it is difficult to parallelize the above code. Iterative approaches can be easily parallelized.

Iterative Approach of Cooley-Tukey FFT:

```
algorithm iterative-fft is
   input: Array a of n complex values where n is a power of 2
   output: Array A the DFT of a

   bit-reverse-copy(a,A)
   n ← a.length
   for s = 1 to log(n)
       m ← 2^s
       ω_m ← exp(−2πi/m)
       for k = 0 to n-1 by m
           ω ← 1
           for j = 0 to m/2 – 1
               t ← ω A[k + j + m/2]
               u ← A[k + j]
               A[k + j] ← u + t
               A[k + j + m/2] ← u – t
               ω ← ω ω_m

   return A
```

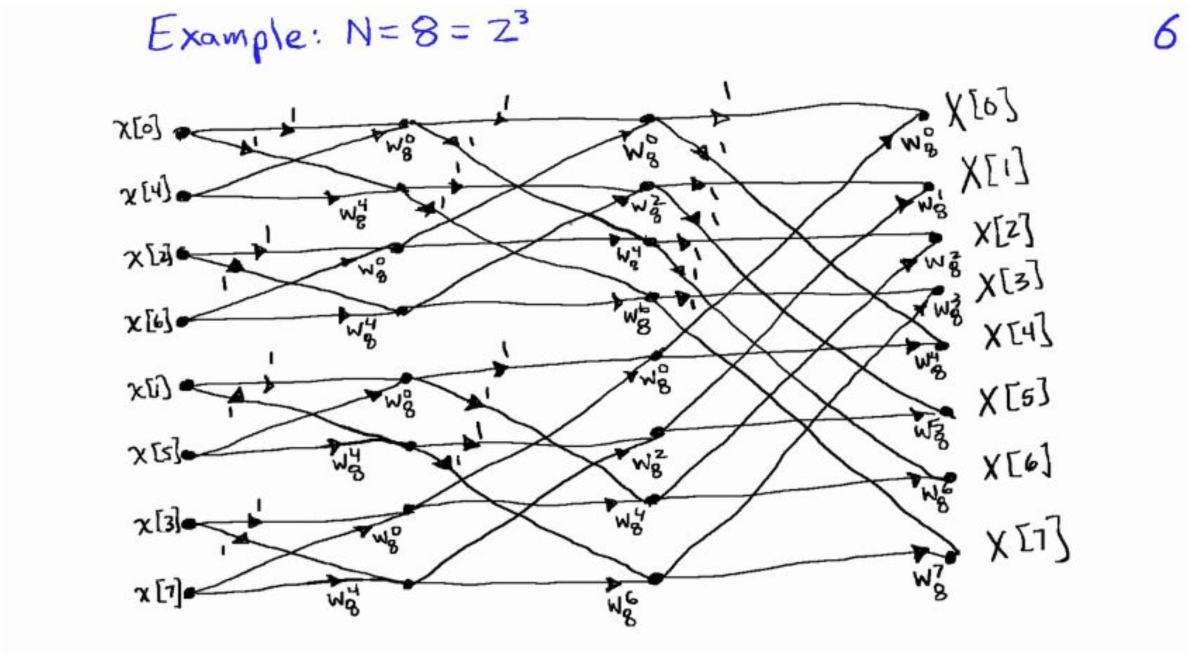
In this method, we are doing bit-reverse copy, pseudocode for this is :

```
algorithm bit-reverse-copy(a,A) is
  input: Array a of n complex values where n is a power of 2,
  output: Array A of size n

  n ← a.length
  for k = 0 to n – 1
      A[rev(k)] = a[k]
```

This method can be easily parallelized as it is an iterative approach.

Butterfly representation of above approach is:



Comparison:

For N = 131072

- For recursive code, time taken is 0.025 sec
- For iterative code, time taken is 0.021 sec
- For omp code,
    - Threads = 4, static schedule, time taken is 0.017 sec
    - Threads = 4, guided schedule, time taken is 0.014 sec
    - Threads = 6, static schedule, time taken is 0.016 sec
    - Threads = 6, guided schedule, time taken is 0.013 sec
    - Threads = 2, static schedule, time taken is 0.019 sec
    - Threads = 2, guided schedule, time taken is 0.017 sec

For N = 262144

- For recursive code, time taken is 0.030
- For iterative code, time taken is 0.024
- For omp code,
    - Threads = 4, static schedule, time taken is 0.019
    - Threads = 4, guided schedule, time taken is 0.016
    - Threads = 6, static schedule, time taken is 0.017

- Threads = 6, guided schedule, time taken is 0.015
- Threads = 2, static schedule, time taken is 0.022
- Threads = 2, guided schedule, time taken is 0.020

Observation

- From the above experiment, we can observe that the time taken by recursive approach is more when compared to iterative approach
- Also, we can observe that the Parallelised iterative approach is faster than other two methods.
- We can also observe that if we increase the threads, speed of execution also increases.
- Also we can observe that, guided works better than static.