COMPUTATION OF THE DISCRETE FOURIER TRANSFORM - PART 3

Solution 20.1

	6.10	6.12	6.13	6.14	6.18	6.21	6.22	6.23
(1)	yes	yes	no	no	yes	yes	no	no
(2)	no	yes	yes	no	yes	no	yes	yes
(3)	yes	no	yes	yes	no	yes	yes	no
(4)	no	yes	no	no	no	yes	no	no
(5)	no	no	no	yes	no	no	no	yes

Solution 20.2

(a) As discussed in the lecture, the coefficients in the inverse DFT are the complex conjugates of the coefficients in the DFT. In addition the inverse DFT has associated with it a scale factor of 1/N. Consequently to modify any of the radix-2 algorithms to implement an inverse DFT, we replace the coefficients by their complex conjugates and apply a scale factor of (1/N) to the output. An alternative is to use the result of problem 2 lesson 18.

(b) and (c) The two algorithms which are particularly convenient when data is to be stored in and accessed from sequential memory such as disk are represented by the flow-graphs of Figures 9.16 and 9.24 of the text. Since the flow-graph of Figure 9.24 has data in normal order as its input, and the flow-graph of Figure 9.16 of the text has data in normal order as its output , it is most convenient to use the flow-graph of Figure 9.24 for the computation of the DFT and the flow-graph of Figure 9.16 for the computation of the inverse DFT.

(d) In the algorithm of Figure 9.24, successive points in an array are computed by combining data from the first half and last half of the previous array. Consequently the first half of the input data should be stored on track A and the second half on track B. In computing the first array, the first half of the computed points are stored in track C and the second half on track D. In computing the second array we combine the data on tracks C and D, storing the first half of the results on track A and the second half on track B.

(e) There are $\log_2 N$ arrays to be computed. Thus, if $\log_2 N$ is even, the results are on tracks A and B. If $\log_2 N$ is odd, the results are on tracks C and D. Furthermore, the DFT values are stored on these tracks in bit-reversed order. For $\log_2 N$ even, the even numbered points are in bit-reversed order on track A and the odd numbered points are in bit-reversed order on track B.

(f) The algorithm of Figure 9.16 which is to be used for the inverse transform assumes that the input is in bit-reversed order. Consequently it is not necessary to sort the DFT values into normal order. In the algorithm of Figure 9.16 the computation of an array involves combining successive points in the previous array to obtain results in the first half and last half of the array being computed. Assume that the results from the computation of the DFT are stored on tracks A and B. As discussed in (e), the first half of these points (the even numbered points) are on track A and the second half (the odd numbered points) are on track B.

In implementing the flow-graph of Figure 9.16 we first combine successive points from track A generating results for the first and last half of the first array. When track A is completed, we combine successive points from track B. Thus in using the output of the flow-graph of Figure 9.24 as the input to the flow-graph of Figure 9.16 no rearrangement of data is necessary. The way in which data is accessed and stored is however different for the two algorithms.

Solution 20.3

Let m denote a memory location ($0 \leq m \leq N - 1$) and $\hat{m}$ the corresponding bit-reversed location. According to the flow-chart of Figure P20.3-2 when counter A is equal to m, the data in locations m and $\hat{m}$ are interchanged and when counter A is equal to $\hat{m}$ the data in locations m and $\hat{m}$ are again interchanged. Consequently at the end of the program in Figure P20.3-2 the data is still in normal order. To correct this it is necessary to ensure that an exchange between a memory location and its bit-reversed counterpart is made only once. One possible correction to the flow-chart of Figure S20.3-1 is indicated in Figure S20.3-1. This correction also takes into account the obvious fact that when $m = \hat{m}$ no exchange is necessary.

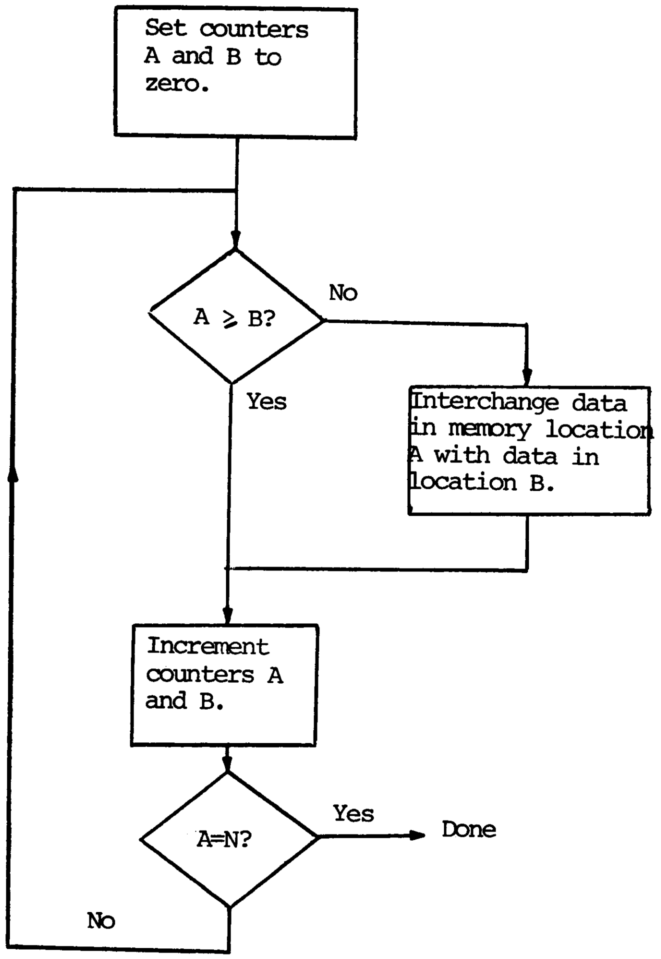
Set counters A and B to zero.
A ≥ B?
No
Yes
Interchange data in memory location A with data in location B.
Increment counters A and B.
A=N?
Yes
Done
No

Figure S20.3-1

Solution 20.4

With N = 9 we divide x(n) into three subsequences each containing three points as indicated in Figure S20.4-1

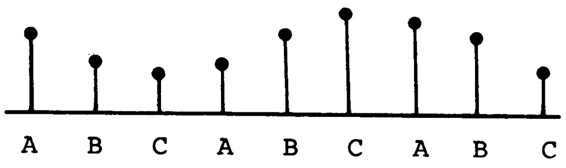

Figure S20.4-1

Thus,

$$X(k) = \sum_{n=0}^{8} x(n)W_9^{nk} = \underbrace{\sum_{r=0}^{2} x(3r)W_9^{3rk}}_{\text{Subsequence A}} + \underbrace{\sum_{r=0}^{2} x(3r+1)W_9^{(3r+1)k}}_{\text{Subsequence B}}$$

$$\underbrace{+\sum_{r=0}^{2} x(3r+2)W^{(3r+2)k}}_{\text{Subsequence C}}$$

Using the fact that $W_9^{\,3} = W_3$

$$X(k) = \sum_{r=0}^{2} x(3r)\; W_3^{\,rk} + W_9^{\,k}\sum_{r=0}^{2} x(3r+1)\; W_3^{\,rk} + W_9^{\,2k}\sum_{r=0}^{2} x(3r+2)\; W_3^{\,rk}$$

The resulting flowgraph is shown in Figure S20.4-2.

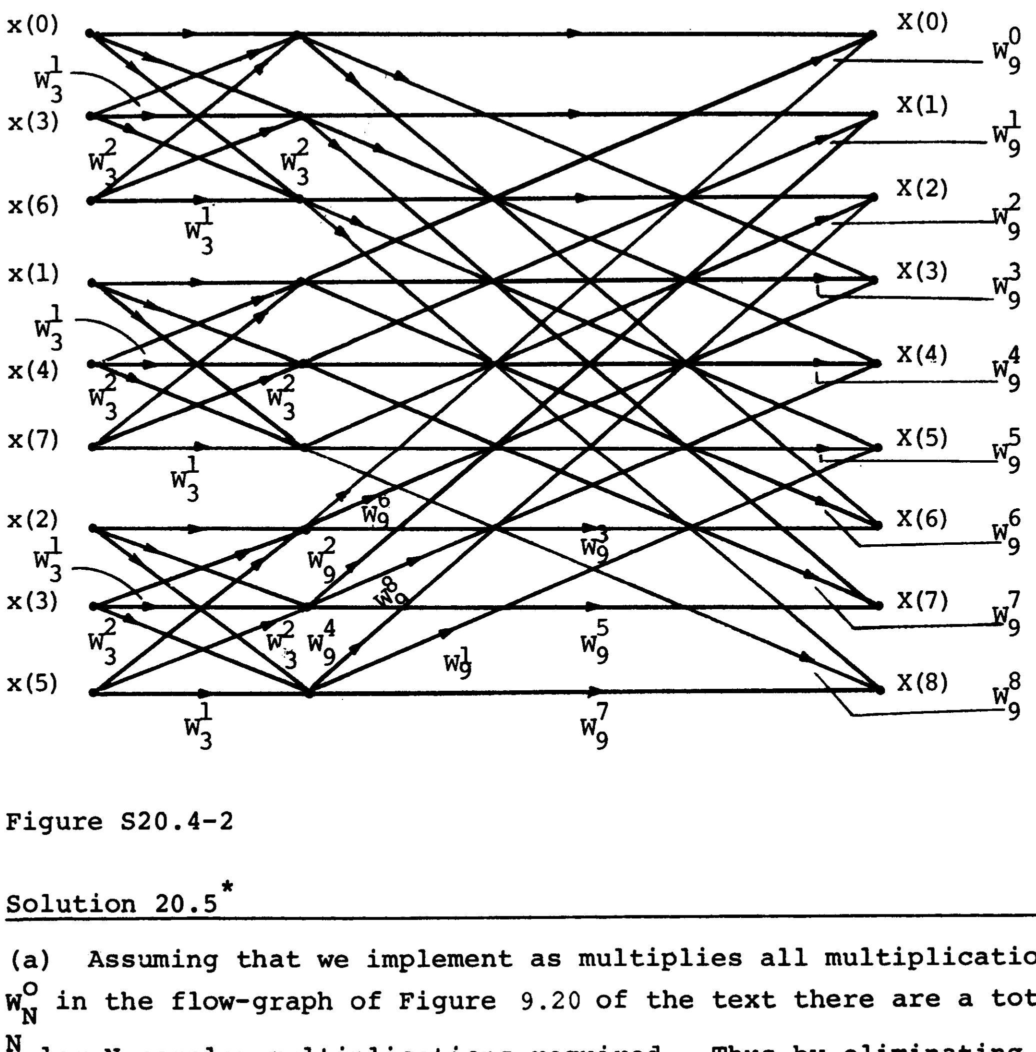

Figure S20.4-2

Solution 20.5*

(a) Assuming that we implement as multiplies all multiplications by W_N^0 in the flow-graph of Figure 9.20 of the text there are a total of $\frac{N}{2}\log_2 N$ complex multiplications required. Thus by eliminating $\frac{N}{2}$ complex multiplications the percentage reduction is

$$\frac{(N/2)}{(\frac{N}{2})\log_2 N} \cdot 100\% = \frac{1}{\log_2 N} \cdot 100\%$$

(b) Change line 5 to DO 20 L = 1, M - 1

insert after line 16:

```
      DO 22 I=1,N,2
      IP=I+1
      T=X(I)+X(IP)
      X(IP)=X(I)-X(IP)
22    X(I)=T
```

(c)

```
      SUBROUTINE FFT(XI,XR,M)
      DIMENSION XI(1024), XR(1024)
      N=2**M
      PI=3.14159265358979
      DO 20 L=1,M
      LE=2**(M+1-L)
      LE1=LE/2
      UR=1.0
      UI=0.0
      WR=COS(PI/FLOAT(LE1))
      WI=-SIN(PI/FLOAT(LE1))
      DO 20 J=1,LE1
      DO 10 I = J,N,LE
      IP=I+LE1
      TR=XR(I)+XR(IP)
      TI=XI(I)+XI(IP)
      TMR=XR(I)-XI(IP)
      TMI=XI(I)-XI(IP)
      XR(IP)=TMR*UR-TMI*UI
      XI(IP)=TMR*UI+TMI*UR
      XR(I)=TR
10    XI(I)=TI
      TR=UR*WR-UI*WI
      UI=UR*WI+UI*WR
20    UR=TR
      NV2=N/2
      NM1=N-1
      J=1
      DO 30 I=1,NM1
      IF(I.GE.J) GO TO 25
      TR=XR(J)
      TI=XI(J)
      XR(J)=XR(I)
      XI(J)=XI(I)
      XR(I)=TR
      XI(I)=TI
25    K=NV2
      IF(K.GE.J) GO TO 30
      J=J-K
      K=K/2
      GO TO 26
30    J=J+K
      RETURN
      END
```