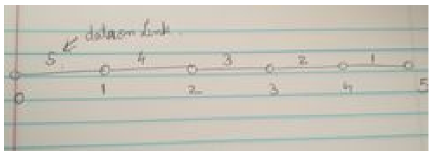
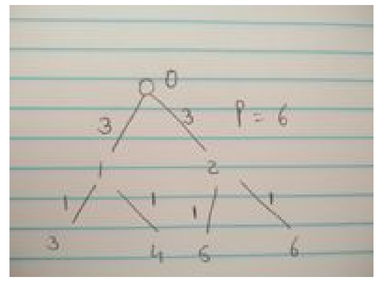
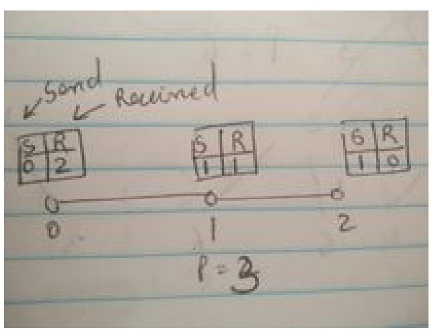
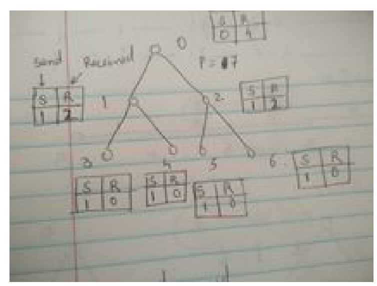
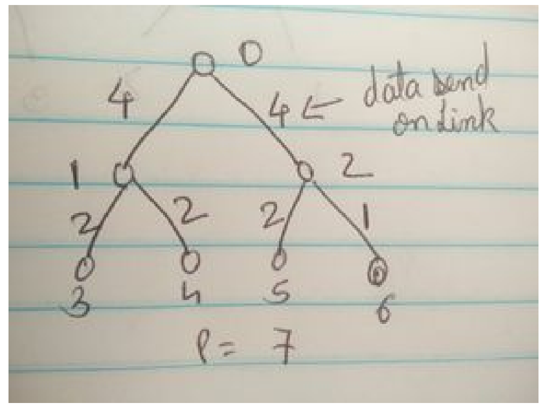
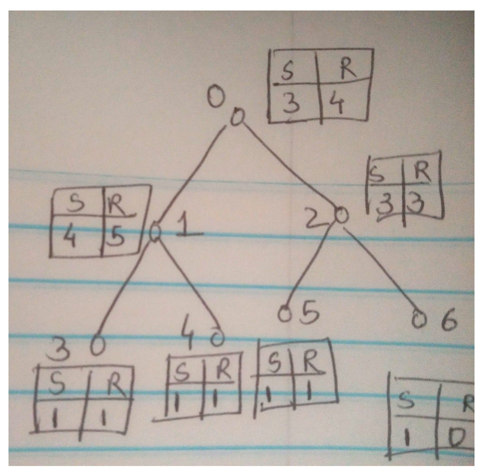
Assignment 6: distributed memory basics
By Abhishek Bhandwaldar
Q1.

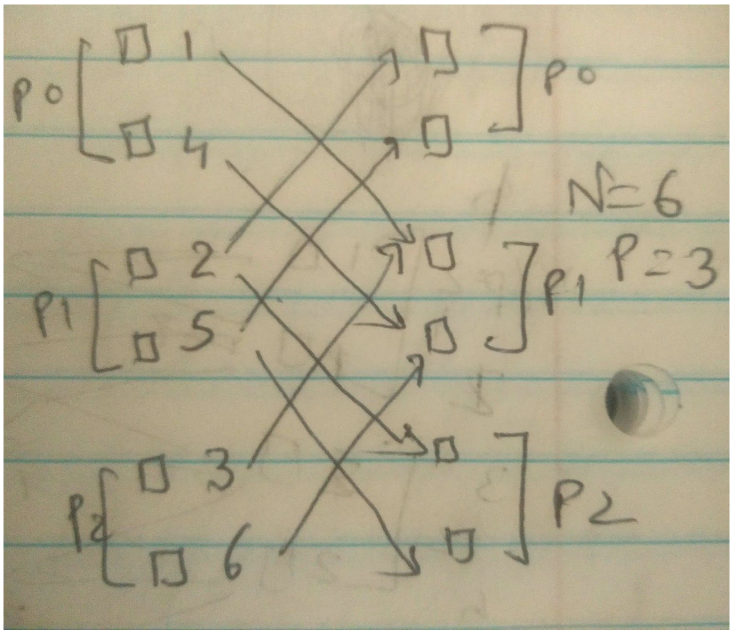
	Algorithm 1 Network Structure 1	Algorithm 1 Network Structure 2	Algorithm 1 Network Structure 3
Data on Each Link	Data on each link in total: 	Data on each Link in total: Data is 1.	Data on each link in total: 
Most Loaded Link	From above the Link 0 – 1 is most loaded	Every link has same load hence no one link with highest load.	Link 0-2 and Link 0-1 are most loaded.
Data Send/Received by Each Node		0^{th} node receive p-1 data and sends 0 data other nodes receive 0 data and send 1 data	
Longest Chain of communication	Longest chain of communication is between last node in chain and 0^{th} node.	All communication are same.	Longest communication is between the leaf node of tree and root node of tree.

	Algorithm 2 Network Structure 1	Algorithm 2 Network Structure 2	Algorithm 2 Network Structure 3
Data on Each Link	Data send on each link is 1	Data Send on each link is 1	Data send on each link differs with its depth in tree. 
Most Loaded Link	All links have same load	All Links have same Load	Link 0-1 and 0-2 are have highest load
Data Send/Received by Each Node	Except for last node and 0th node every node receives and send 1 data. 0th node receives 1 data and last node sends 1 data	Except for last node and 0th node every node receives and send 1 data. 0th node receives 1 data and last node sends 1 data	Assumption: I have considered send and receive of intermediate nodes also while sending from 1 node to another and there is no direct link. 
Longest Chain of communication	There is no longest chain of communication.	There is no longest chain of communication.	Longest chain of communication is between node from 1 subtree to another. In above diagram it from node 5 to node 4

	Algorithm 3 Network Structure 1	Algorithm 3 Network Structure 2	Algorithm 3 Network Structure 3
Data on Each Link	Algorithm Not correct	Algorithm Not correct	Algorithm Not correct

Q2:

1. Algorithm for round robin:



a. Initialize k=0
b. Repeat below steps for all values of k
c. If k=0
    i. For every element send it to p-1 and p+1. If p=0 only send to p+1 and if p=P-1 then only send to p-1
    ii. k++
d. Else
    i. For every element receive data from p-1 and p+1. If p=0 only receive to p+1 and if p=P-1 then only receive to p-1
    ii. Calculate Heat equation. Using below equation:

$$Heat^k[0] = \frac{2Heat^{k-1}[0] + Heat^{k-1}[1]}{3}$$

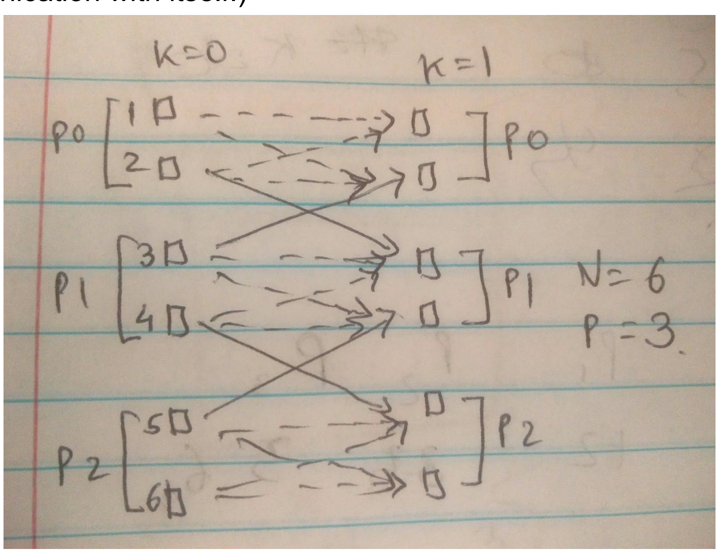
$$Heat^k[n-1] = \frac{2Heat^{k-1}[N-1] + Heat^{k-1}[N-2]}{N-2}$$

$$Heat^k[i] = \frac{Heat^{k-1}[i-1] + Heat^{k-1}[i] + Heat^{k-1}[i+1]}{3}, \forall 0 < i < N-1$$

iii. For every element send it to p-1 and p+1. If p=0 only send to p+1 and if p=P-1 then only send to p-1

iv. k++

2. Algorithm for block: (Solid lines is communication between processors and dashed line is communication with itself.)



a. Initialize k=0
b. Repeat below steps
c. If k = 0

        i. Send the first and last element of the data allocated to that processor to p-1 and p+1. In case p is 0th processor only send to p+1. Eg : in figure p1 will send the 3rd and 4th element to p0 and p2
        ii. k++
    d. Else
        i. Receive data from p-1 and p+1. If p=0 then only receive from p+1 and if p = P-1 then only receive from p-1.
        ii. Calculate Heat[i] for all the elements allocated to that processor. Eg for element 5 and 6 for processor p2. Using below

$$Heat^k[0] = \frac{2Heat^{k-1}[0] + Heat^{k-1}[1]}{3}$$

$$Heat^k[n-1] = \frac{2Heat^{k-1}[N-1] + Heat^{k-1}[N-2]}{N-2}$$

$$Heat^k[i] = \frac{Heat^{k-1}[i-1] + Heat^{k-1}[i] + Heat^{k-1}[i+1]}{3}, \forall 0 < i < N-1$$

        iii. Send the first and last element of the data allocated to that processor to p-1 and p+1. In case p is 0th processor only send to p+1.
        iv. k++
        v. Goto step 2. Repeat for all values of k.

3. Communication per iteration is 2. As send and receive between same processor is not counted.
4. Block as it uses 2 communication per node whereas round robin uses N/P *2 communication per node. So we can use any one of them. But block communication is easy whereas round robin is tricky.

Round robin:
Total Communication: N/P *2 per node except for first and last which is N/P*1. Total is (P-2)*N/P*2 + 2*N/P*1.
Per link: N/P*2 per link. As n figure where N/P =2 communication between p2 and p1 happens 2 times and between p1 and p0 happens 2 times. Ie 4 times in total.
Per Node: Each node receives N/P*2 data except for first and last node.

Block:
Total Communication: 2 per iteration per node and 1 for first and last node. Therefore total is (P-2)*2 + 2 per iteration.
Per link: 2 as for example in above figure p2 send to p1 and p1 sends to p2 on same link.
Per Node: Each node receives 2 data and sends 2 data except for first and last node which each send 1 data.

Q3.

1. Assumption : x is a column vector.

a. Algorithm for horizontal decomposition:
   i. The matrix are decomposed horizontally. Hence each processor receives N/P rows of matrix A and N/P elements of column vector x.
   ii. So each processor has partial copy of vector x.
   iii. Send the value of column vector x to every processor in network.
   iv. For i=p*N/P to (p+1)*N/P do
      1. Send the x [i] to every processor (Scatter)
   v. For i=0 to (p)*N/P and i = (p+1) * N/P to N do
      1. Receive The x [i] to from every processor in network (Gather)
   vi. Now each processor has its own copy of x.
   vii. Compute Matrix multiplication between the N/P rows (p*N/P to (p+1)*N/P) at each processor and Column vector x and save the result in y.
   viii. Copy the result in variable x.
   ix. Goto to step 3. Repeat above for 10 iterations.

b. Algorithm for Vertical decomposition:
   i. The matrix A and column vector x are decomposed vertically. Every processor will receive N/P columns of matrix A and processor 0 will receive the complete vector x as it is only 1 column wide.
   ii. If p==0 (if it is 0th processor)
      1. Send column vector x to every other processor in network such that every processor receives N/P block. (Scatter). Eg: 2nd processor receives 2nd N/P blocks 3rd receives 3rd N/P blocks and so on.
   iii. If p!=0
      1. Receive value of column vector x
   iv. For row = 1 to N do
      1. For Column= p*N/P to (p+1)N/P do
         a. Multiply a[row][Column] * x[Column] and store in data vector.
   v. For i= 1 to N do and i != p
         a. Send data vector to i
   vi. For i= 1 to N do and i != p
         a. Receive data vector from other processors and aggregate to vector y
   vii. Copy value of y into x and Goto step 4. Repeat for 10 iterations.

c. Algorithm for Block decomposition:
   i. Every processor has block of N/sqrt(P) * N/sqrt(P) block. Every processor has N/sqrt(P) rows of the column vector x
   ii. Every processor will compute N/sqrt(P) rows of solution vector y.

   iii. For row = p%sqrt(P) * N/sqrt(P) to (p+1)%sqrt(P)* N/sqrt(P)

        1. For column= p%sqrt(P) * N/sqrt(P) to (p+1)%sqrt(P)* N/sqrt(P)
            a. y[row] = y[row] + A[row][column] * x[column]
    iv. If p is not diagonal element in processor network than
        1. Send its local Y vector to closest diagonal numbered processor. Eg. In P=9 processor network , p2 will send to p0, Processor p4 will send to itself and processor p7 will send to p8

p0	p1	p2
p3	p4	p5
p6	p7	p8

    v. If processors diagonal element in network than it will receive all the y vectors and aggregate it to create single vector.
    vi. Copy y into x and send x to all processor in same row. Eg p4 will send to p3 and p5
    vii. Goto step 3 and repeat above steps for 10 iterations.

2. Memory needed: Consider each element requires 1 unit of memory
    a. Horizontal Decomposition: N^2/P + N units
    b. Vertical Decomposition: N^2/P + N units
    c. Block Decomposition: N^2/P + N units
3. Communication:
    a. Horizontal:
        i. Total: P* (P-1) * N/P
        ii. Per link: P-1 (every processor sends to every other processor)
        iii. Per Node: P (every Processor receives P-1 and sends 1 data to P-1 processors)
    b. Vertical:
        i. Total: P* (P-1) * N/P
        ii. Per link: P-1 (every processor sends to every other processor)
        iii. Per Node: P (every Processor receives P-1 and sends 1 data to P-1
    c. Block:
        i. Total: (P-sqrt(P)) * N/sqrt(P)
        ii. Per link: N/sqrt(P)
        iii. Per Node: diagonal receives input from all processor in row.