QUESTION POOL (CSc 387)

I. In this section, there are XX TRUE/FALSE questions each worth Y points. For each question, please circle one of (T)rue or (F)alse.

T **F** If manufacturers of massively parallel computers are increasing the top performance of their systems by a factor of 128 every 126 months, then it is fair to say that, on the average, the performance of supercomputers are doubling every 18 months.

T **F** Amdahl's law can be rephrased as in the following:
"The maximum speedup obtainable from a parallel program is proportional with the product P*k where P is the number of processors used and k is the percent time needed to run sequentially the program instructions that can be perfectly parallelized."

T **F** Gustafson's Law states that the maximum speedup can go as high as (s + P*(1-s)) where (P = number of processors) and (s = the fraction of time needed to run "non-parallelizable" operations in the program using P processors)

T **F** Consider a parallel program which requires communication among the processors and has a constant number of parallel tasks to perform. If we assume that the total work is evenly divided among parallel processors and the best effort is done to increase the efficiency, it is always true that the efficiency gets smaller as we increase the number of processors.

T **F** Consider a k-stage pipeline. If the slowest stage takes as much time to compute as the sum of the computation times of all the other (k-1) stages, then the maximum speedup can not exceed 2.

T **F** Under the ideal conditions, a k-stage pipeline can obtain a speedup of n while processing an input vector of n elements.

T **F** Under the ideal conditions, a k-stage pipeline can obtain a speedup of at most k.

T **F** If sending a unit length message takes one unit time for a processor, one-to-all personalized communication (scatter) on a hypercube of N processors can be done in O(logN) time.

T **F** In a systolic computer, the computing cells can perform different functions on different data.

T **F** The diameter of a full-binary-tree network with $(2^n - 1)$ processors is 2(n-1)

T **F** There are no cycles of odd length in a hypercube of dimension d.

T **F** We can embed a ring of size P=27 onto a 5-dimensional hypercube.

T **F** There are $C\begin{pmatrix} n \\ d \end{pmatrix}$ processors which are d distance away from any given processor in an n-dimensional hypercube.

T **F** On a n-dimensional hypercube, average distance between any two processors is n/2.

T **F** For a n-dimensional hypercube topology, the bisection width is 2^n

T **F** If there are N tasks and P processors in a parallel system, potentially, there are P^N different ways that we can run N tasks on P processors. Therefore, finding the *global optimum* task assignment is an NP-Complete problem in general.

T **F** If, in the future, it becomes feasible to change the number of processors in a parallel system dynamically as a <u>linear</u> function of the input size (N), then we can solve sequentially intractable problems (in the NP-Complete class) in polynomial time.

T **F** Power-shift (shift amount is a power of 2) can be done in O(1) in a hypercube.

T **F** On a 1024 processor MIMD hypercube, a circular shift of $(2^i - 2^j)$ where i and j are known apriori can be performed in 4 steps using the gray code ordering.

T **F** On a P processor MIMD hypercube, a circular shift of k can be performed in 4 steps using the gray code ordering.

T **F** On a P processor MIMD hypercube, a circular shift of k can be performed in at most $2log_2k$ steps using the gray code ordering.

T **F** On a P processor MIMD hypercube, a circular shift of k **can not** be performed in less than $O(k)$ steps using the gray code ordering.

T **F** First N elements of a Fibonacci sequence can be computed in $O(logN)$ time in parallel on a hypercube of N processors.

T **F** (000, 001, 010, 011, 100, 101, 110, 111) is a gray code.

T **F** (000, 001, 011, 010, 110, 111, 101, 100) is a gray code.

T **F** If we compare a hypercube with a full-binary-tree network each having O(2^n) processors where $n >$ 3, hypercube is always superior (preferred) to full-binary-tree in terms of both *bisection width* and *diameter.*

T **F** A call to MPI_Barrier() blocks the calling process until all the processes in the communicator have reached this routine. Therefore, it may introduce delays in a program.

T **F** It is claimed that MPI is portable. Then, it should be true that we can write MPI programs which can run on both distributed and shared memory machines.

T **F** Processes participating in an MPI_Barrier() call will deadlock if one of the processors in the communicator never makes the call.

T **F** Consider an MPI program running on a network of workstations and each process is running on a separate processor. You can declare a global array which is shared among all the processes and accessed by each process without any explicit message passing.

T **F** If sending a message of size one unit length takes one unit time for a processor, global broadcast operation (one-to-all) can not be completed less than $logP$ time on any parallel system with P processors. (Assume that there is no broadcast bus available and, at any time, a processor can communicate with at most one other processor).

T **F** All-to-all broadcast of a unit-length message can be done in O(logP) time on a hypercube with P processors.

T **F** All-to-all broadcast of a unit-length message can be done in O(P) time on a hypercube with P processors.

T **F** All-to-all broadcast of a unit-length message can be done in O(P) time on a ring of P processors.

T **F** On an irregular interconnection network with bidirectional static links, if a certain node can broadcast in time $O(f(n))$, then, theoretically, any node can broadcast in $O(f(n))$ time (even though there is no guarantee that the algorithm will not be more complex).

T **F** *Cache coherence problem* occurs when there are multiple distributed caches on a shared memory architecture.

T **F** Message-passing architectures don't have *cache coherence problem.*

T **F** There are 10 kinds of people in this world, those who understand binary and those who don't!

T **F** The only difference between systolic and pipelined computing is that systolic computers have identical functional units while pipelined arrays may, in general, have different functional stages.

T **F** Logically speaking, a parallel program designed to run on an SIMD computer can be easily rewritten to run on an MIMD computer without causing much change to the run-time efficiency. However, the opposite is not necessarily true.

T **F** If (N >> P), it is possible to achieve asymptotically linear speedup while adding N numbers using P processors.

T **F** If (N = P), it is possible to achieve asymptotically linear speedup while adding N numbers using P processors.

T **F** The success of Monte Carlo methods critically depend on two factors: (i) a good random number generator which allows random selections in calculations, and (ii) a large number of such calculations

T **F** If the processors of a multicomputer are fully interconnected (complete graph) in a pairwise manner, a load balanced mapping (equal load on each processor) is sufficient to optimize the running time of a parallel program. Assume that (# of Tasks >> # of PEs).

T **F** Consider solving an lower triangular system of linear equations with N unknowns. If we use the pipelining approach decribed in class with N pipeline stages, parallel running time, T_{par}, is O(N) in the asymptotic case (for very large N) while the sequential running time is O(N^2).

T **F** A "good" random number generator is given as:

$$x_{i+1} = (ax_i + c)\ MOD\ m$$

This function generates a repeating sequence of $(2^{31} - 2)$ different random numbers when $a = 16807$, $m = 2^{31} - 1$, and $c = 0$. In order to generate *uncorrelated* random numbers in parallel on P processors, all we need to do is, to start with a different seed (selected randomly) in each processor and let each processor use the same function above.

T **F** Given N processes ranked from 1 to N, they can compute (N!) using MPI_Reduce().

T **F** *Latency Hiding* is a technique to overlap communication with computation in order to increase the speed and efficiency of a parallel program. Typically, it involves using **blocking** send/receive routines.

(T/F): END OF TEST I

T **F** It is possible to obtain linear speedup when sorting N^2 numbers on a regular NxN square mesh of processors (Note that the speedup is calculated with respect to the time complexity of the best sequential sorting algorithm).

T **F** Any sorting algorithm that can sort all 0/1 sequences correctly can also sort all arbitrary sequences.

T **F** The following sequence is bitonic: 11 13 19 19 14 10 8 9 9

T **F** If a list of N numbers is known to be **bitonic**, it can be sorted in $O(\log N)$ time on a hypercube of N processors.

T **F** **Odd-even transposition sort** algorithm is cost-optimal when P = O($\log N$) where P is the no. of processors and N is the input size.

T **F** RankSort sorts N^2 numbers in $O(\log N)$ time on a hypercube of $P = N^2$ processors

T **F** Shearsort sorts N^2 numbers in $O(N \log N)$ time on a **mesh** of NxN processors

T **F** Among the sorting algorithms; *bitonic mergesort, rank sort, and shearsort*; *rank sort* runs the slowest using P=n processors to sort n numbers (assume that a network topology is selected to optimize the running time).

T **F** **Odd-even transposition sort** algorithm never takes more than N steps to sort N elements using P=N processors.

T **F** **Bitonic sort** algorithm can sort N numbers in O($log^2 N$) time on a hypercube of P=N processors.

T **F** **Bitonic sort** algorithm can sort N numbers in O($logN$) time on a hypercube of P=N processors.

T **F** Consider a hypercube with P processors. We logically partition the hypercube into k windows of size W each (each window forms a subcube and there are W processors in each window). If each window broadcasts an item to the processors in that particular window, all of the broadcasts (a total of k broadcasts in k windows) can complete in O(logW) time.

T **F** Graph partitioning problem as explained in class is in NP-Complete class. Hence, heuristics are used to find suboptimal solutions.

T **F** If we bitwise-XOR both X and Y with a number N to obtain $\hat{X} = X \oplus N$ and $\hat{Y} = Y \oplus N$, the Hamming Distance between $\hat{X}$ and $\hat{Y}$ will be the same as that of X and Y.

T **F** There is a dilation 1 embedding of T_3, a complete binary tree with 7 nodes into H_3, a 3-dimensional hypercube.

T **F** If we are allowed to use as many processors as we need, we can multiply two NxN matrices in O(1) time on a hypercube multicomputer.

T **F** Consider the parallel multiplication of two NxN matrices on a 2D Mesh of $P = N^2$ processors using Cannon's Algorithm. Assume that, initially, corresponding elements of A and B are stored in N^2 processors, one element from each matrix per processor. Asymptotically speaking, this algorithm is **cost-optimal**.

T **F** Consider the task and data partitioning for Parallel Gaussian Elimination (as in project #4). If $N >> P$, the best approach to balance the workload on each processor is to assign N/P <u>consecutive</u> columns (rows) to each processor.

T **F** Genetic algorithms are guided random search techniques based on the mechanics of biological evolution.

T **F** For a given problem, results obtained through a genetic algorithm are always and consistently better than those obtained by an algorithm which uses a totally random search.

T **F** If we are allowed to use as many processors as needed, multiplication of two NxN matrices can be done in $O(logN)$ time on a hypercube.

T **F** Multiplication of two (NxN) matrices can be performed in $\leq (N+1)$ steps on a 2-D systolic array with N^2 processing elements (cells).

T **F** In Gaussian Elimination procedure, if *partial pivoting* is used to get the i^{th} pivot, then we swap the i^{th} row with the row below it that has the largest absolute element in the i^{th} column of any of the rows below the i^{th} row if there is one.

T **F** Jacobi iterative technique for solving large systems of equations in the form $Ax = b$ is gauranteed to converge if A a *symmetric* matrix.

T **F** Jacobi iterative technique for solving large systems of equations in the form $Ax = b$ is gauranteed to converge if A is a *diagonally dominant* matrix.

T **F** Consider the parallel implementation of Dijkstra's Single-Source Shortest Paths (SSSP) algorithm. It can be implemented cost-optimally if $\frac{p \log p}{n} = O(1)$ where p is the number of processors and $n = |V|$.

(T/F): END OF TEST II

II. In this section, there are XX multiple choice questions each worth YY points.

1. Suppose the following two processes are running in a shared memory environment.

PROCESS-A	PROCESS-B
...	...
X := 3	X := 0
X := X + 1	X := 2*X
...	...

After both A and B terminate, it is **not possible** that the shared variable X take a value of:
(a) 8 (b) 0 (c) 7 (d) 6 (e) 2

2. Among the network parameters (criteria) listed below, which of the following are desired to be "higher" in value? Circle all of those that apply.
(a) Diameter (b) Bisection width (c) Bandwidth (d) Network latency (e) Cost

3. Which of these interconnection networks would be the easiest to scale to large sized networks without making significant changes to the hardware design of the processor and the network?
(a) Hypercube (b) crossbar (c) 2-D Mesh (d) Completely connected graph

4. How many nodes are there which are 2 distance away (considering only shortest routes) from a given node in a hypercube of dimension n ?
(a) $\frac{n(n-1)}{2}$ (b) n (c) $2n$ (d) $C\binom{n-2}{2}$ (e) none of these

5. How many different Gray-Code sequences can be generated using the processor labels of an hypercube which has $N = 2^d$ number of processors ?
(a) N (b) N ! (c) $(\log d)!$ (d) $(\log N)!$ (e) d

6. Which one of the following is not true for a d-dimensional hypercube?

   (a) it has a total of d*2^{d-1} links
   (b) a *ring* of size 2^{d-1} can be embedded into it with dilation 1
   (c) the bisection width is 2^{d-1}
   (d) The diameter is d.
   (e) A mesh of size $m_1 \times m_2$ can be embedded into it with dilation 1 if $(m_1 \times m_2) \leq 2^d$

7. Consider finding all the prime numbers upto N in parallel on P ($\leq N$) processors using the optimized Parallel Prime Number Sieve Algorithm. If k represents the total number of primes between 1 and $\sqrt{N}$, then, for all practical purposes, which one of the following comes closest to the time complexity of this algorithm:
(a) $\frac{N}{P}$ (b) $\frac{N^2}{P} * k$ (c) $N^{3/2}$ (d) $\frac{N}{P} + \sqrt{N}$ (e) $\frac{\sqrt{N}}{P} * k$

8. We would like to embed a (17x5) mesh in a hypercube with dilation one. What is the minimum dimension hypercube needed?
(a) 7 (b) 8 (c) 9 (d) 85 (e) none

9. Suppose that MPI_COMM_WORLD consists of the three processes 0,1, and 2, and suppose the following code is executed:

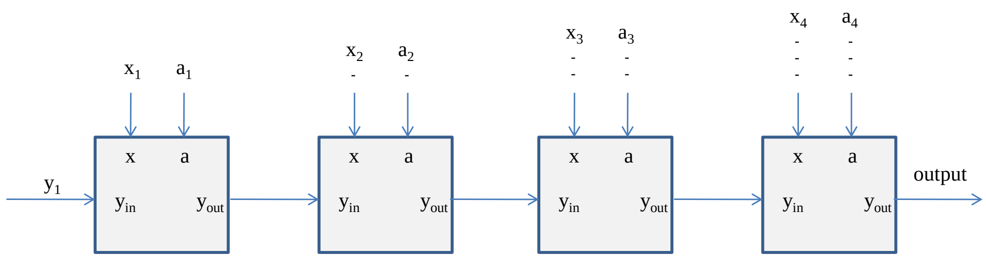
```
int x, y, z;
switch(my_rank) {
 case 0: x=0; y=1; z=2;
    MPI_Bcast(&x, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Send(&y, 1, MPI_INT, 2, 43, MPI_COMM_WORLD);
```

```
     MPI_Bcast(&z, 1, MPI_INT, 1, MPI_COMM_WORLD);
     break;
  case 1: x=3; y=8; z=5;
     MPI_Bcast(&x, 1, MPI_INT, 0, MPI_COMM_WORLD);
     MPI_Bcast(&y, 1, MPI_INT, 1, MPI_COMM_WORLD);
     break;
  case 2: x=6; y=7; z=8;
     MPI_Bcast(&z, 1, MPI_INT, 0, MPI_COMM_WORLD);
     MPI_Recv(&x, 1, MPI_INT, 0, 43, MPI_COMM_WORLD, &status);
     MPI_Bcast(&y, 1, MPI_INT, 1, MPI_COMM_WORLD);
     break;
}
```

What are the values of x, y, and z on each process after the code has been executed?

(a)	(b)	(c)	(d)	(e) none
x, y, z	x, y, z	x, y, z	x, y, z	
p0: 0, 1, 2	p0: 0, 1, 8	p0: 0, 1, 4	p0: 0, 1, 2	
p1: 3, 8, 5	p1: 0, 8, 5	p1: 0, 4, 5	p1: 0, 2, 5	
p2: 6, 7, 8	p2: 1, 8, 0	p2: 1, 4, 0	p2: 1, 2, 0	

10. The pipeline given below consists of four stages and it is synchronous, i.e. each cell finishes its operation in one clock cycle and the (input/output) data advances one step forward.



If each stage performs the operation

$$y_{out} = y_{in} + a.x$$

after 4 clock cycles, the final **output** will be:

(a) $(y_1 + a_1.x_1).(a_2.x_2).(a_3.x_3).(a_4.x_4)$
(b) $y_1 + a_1.x_1 + a_2.x_2 + a_3.x_3 + a_4.x_4$
(c) $y_1.(a_1.x_1 + a_2.x_2 + a_3.x_3 + a_4.x_4)$
(d) $4y_1 + a_1.x_1 + a_2.x_2 + a_3.x_3 + a_4.x_4$
(e) $y_1.(1 + a_1.x_1 + a_2.x_2 + a_3.x_3 + a_4.x_4)$

(Multiple Choice): END OF TEST I

---

11. In a bus-oriented shared memory multiprocessor system, there are P processors connected to the bus and they have no local memory. Assume that, with this current configuration, P processors use up the total (bus and shared memory) bandwidth. If each processor is provided with a local cache memory, by what factor can P (the number of PEs) be increased before the shared memory becomes a bottleneck. Assume that the shared memory bandwidth is B (bytes per second), and the *hit ratio* for a local cache is the same for all caches and is equal to h.

(a) $\frac{1}{h}$ (b) $\frac{B}{h}$ (c) $P * B$ (d) $\frac{1}{1-h}$ (e) B

12. Which of the following problems have non-polynomial time complexities with respect to the inputs given in the parenthesis? Circle all that apply.

(a) Traveling Salesman Problem (TSP) (N = no. of vertices in the graph)

(b) Finding shortest paths between all vertex pairs in a graph (N = no. of vertices in the graph)

(c) N-queens problem (chessboard is NxN)

(d) Partitioning the vertices of a graph into two equal sets such that the number of edges connecting the two sets will be minimum.

(e) Generating all the combinations of N objects in tuples of 3.

13. Which of the following has a polynomial time complexity with respect to the input given in the parenthesis? (Assume that a conventional computer is used)

    (a) Traveling Salesman Problem (TSP) (N = no. of vertices in the graph)

    (b) Partitioning the vertices of a graph into two equal sets such that the number of edges connecting the two sets will be minimum.

    (c) Finding shortest paths between all vertex pairs in a graph (N = no. of vertices in the graph)

    (d) Finding two prime factors of a given number (N = no. of digits in the number)

    (e) Generating all the combinations of N objects in tuples of 3.

    (f) Given N strings, finding the minimal length superstring which embeds all of the N strings.

14. Which of the following is <u>not</u> among the direct methods for solving large system of equations?

    (a) Gaussian Elimination (b) Gauss-Seidel Method (c) LU Factorization
    (d) Gauss-Jordan Elimination (e) None

15. Consider a pixmap containing (2000 x 2000) 24-bit pixels. Suppose each pixel must be operated upon once for each frame and display rate is 60 frames/second. If each pixel operation takes 10^{-7}seconds on a standard workstation, how many parallel workstations are needed to be able to display the image without any interruption? Assume that there is no extra overhead is involved in parallelization. Give an exact answer.

    (a) 24 (b) 150 (c) 8 (d) 1 (e) more info needed

(Multiple Choice): END OF TEST II

(Multiple Choice): ADDITIONAL QUESTIONS

---

1. The total number of subcubes that exist in an n-dimensional hypercube is equal to:

   (a) 2^n (b) $n2^n$ (c) 3^n (d) $n!2^n$ (e) none

2. Which one of the subcubes below does not have any common node with the 2-dim subcube identified by the binary code 11**0 ?

   (a) 1*0*0 (b) *0**0 (c) 11*** (d) **01* (e) 1101*

3. Consider the dataflow diagram (network of butterflies) for parallel FFT algorithm as given in class. Assume that $N = 2^m$, $P = 2^d$, and $m > d$, and the workload is evenly distributed among the processors (i.e. each processor gets N/P consecutive rows of the butterfly network). After how many steps will processors quit communicating with each other and solely depend on their local data?

   (a) m (b) d (c) P (d) m-d (e) none

4. Suppose that, for an (N x N) image, the number of increments for r and θ used in a Hough transform are O(N) and O(1), respectively. Then, parallel Hough transform can be performed in __________ time on a hypercube of N^2 processors.

   (a) O(1) (b) O($logN$) (c) O(log^2N) (d) O($NlogN$) (e) O(N)

III. This section contains XX questions worth a total of YY points. Please write clearly and show your work.

1. Explain how you can embed a ring of N=2^n processors to a hypercube of dimension=n with dilation one. Give an example with N=8.

2. Show the dilation one embedding of a 3x10 mesh onto a hypercube.

3. It's easy to use a gray-code to embed a ring onto a hypercube when the number of elements used is a power of two. Can we embed a ring (with dilation one) which has an odd number of processors? Can we always embed if the ring has an even number of elements (again, with dilation one)? Explain.

4. What is the smallest size hypercube into which a 3D-Mesh of size 2x3x5 can be embedded (with dilation one) ? Show your work.

5. Prove that a hypercube that can embed any $2D$ mesh of size m_1 x m_2 with dilation 1 does not need to have more than $4m_1m_2$ processors.

6. Is it possible to embed a full binary tree of height 2 (7 nodes) on a 3-dimensional hypercube with dilation 1? If your answer is "yes", show the embedding. Otherwise, explain why it is not possible.

7. Prove that there are no cyles of odd length in a hypercube of dimension d.

8. Explain clearly what each one of the following MPI commands does. You need to specify the input and output parameters and what they store before and after the call.

   (a) MPI_Comm_rank (comm, rank)
   (b) MPI_Bcast (buffer, count, datatype, root, comm)
   (c) MPI_IRecv (buf,count,datatype,source,tag,comm, request)
   (d) MPI_Allgather (sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype, comm)
   (e) MPI_Scatter (sendbuf, sendcnt, sendtype, recvbuf, recvcnt, recvtype, root, comm)
   (f) MPI_Reduce (sendbuf, recvbuf, count, datatype, op, root, comm)

9. In *one-to-all personalized communication*, a single processor sends a unique message of size m to every other processor.
   (i) Prove that the lower bound for the time complexity of this operation is $O(m * p)$ where p is the total number of processors in the system.
   (ii) Explain how you would perform *one-to-all personalized communication* on a hypercube multicomputer in $O(m * p)$ time. Draw figures if necessary.

10. (a) Describe a parallel algorithm to find the **factorial** of all numbers between 1 and N in O(logN) steps using N processors. Do not write the code for it. Just work on an example of 8 processors, and show the steps of the algorithm along with the data flow between processors.
    (b) Is this algorithm cost-optimal ? Explain.

11. Objective of a Ranking Algorithm is to assign to each SELECTED processor a rank such that RANK(i) is the number of selected processors with index less than i. Show the steps of a O(logN) hypercube Ranking algorithm in the following example. Compute the running time and show your work.

```
               PE#:      0    1    2    3       4    5    6    7
                        000  001  011  010     110  111  101  100
-----------------------------------------------------------------
Selected:                *         *            *    *         *
RESULT TO BE
OBTAINED:                0         1            2    3         4
-----------------------------------------------------------------
STEP#
-----------------------------------------------------------------
```

```
 0 (initial
-----------------------------------------------------------------
 1
-----------------------------------------------------------------
 2
-----------------------------------------------------------------
 3
-----------------------------------------------------------------
```

12. If prime numbers between 1 and $\sqrt{N}$ are represented as $pr_1, pr_2, pr_3, \ldots pr_k$ in sorted order, derive the exact formula for the time complexity of an optimized parallel algorithm.

13. Fill in the blank boxes in the following table. Assume that there are a total of p processors in each topology. 2-D Mesh has $p = \sqrt{p}.\sqrt{p}$ processors and p is a power of 2 for hypercube.

Topology	**Diameter**	**Bisection Width**
Completely-connected	1	$(\frac{p}{2})^2$
Complete binary tree	$2(log(p+1)) - 2$	1
Ring	$\frac{p}{2}$	2
2-D Mesh	$2(\sqrt{p} - 1)$	$\sqrt{p}$
2-D Torus	$\sqrt{p} - 2$	$2\sqrt{p}$
Hypercube	$log_2 p$	$\frac{p}{2}$

14. (a) (X points) Show the steps of a pipelined sorting algorithm which sorts the following list in ascending order in less than 10 steps using 5 pipelined stages.
    List to be sorted (rightmost number, 8, is input first): **10, 3, 7, 5, 8**
    (b) (X points) Explain what computations are performed at each stage.
    (c) (X points) What is the time complexity of sorting N numbers using N pipeline stages in this fashion?

15. The Fibonacci sequence is defined as follows:

$$f_0 = 0; \quad f_1 = 1; \quad and \quad f_i = f_{i-1} + f_{i-2} \quad i = 2, 3, \ldots, N$$

Describe an $O(logN)$ algorithm for computing all N elements of the above sequence in parallel on a hypercube of N processors. You need to explain clearly all of the major steps in your algorithm. (Note that using the closed form formula to compute the Fibonacci numbers is not acceptable)

$$F_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad and \qquad F_i = \begin{pmatrix} f_i \\ f_{i-1} \end{pmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} f_{i-1} \\ f_{i-2} \end{pmatrix} = AF_{i-1} \qquad \forall i = 2, 3, \ldots N$$

```
PE#:       0      1      2      3      4      5      6      7
          000    001    011    010    110    111    101    100    |   TIME
                                                                  | COMPLEXITY
-------------------------------------------------------------------------------
```

Init State F_1 A A A A A A A

Step 1: **Processors execute Parallel Prefix Multiply** **O(log N)**

Shift-1 & Mult:	F_1	AF_1	A^2	A^2	A^2	A^2	A^2	A^2
Shift-2 & Mult:	F_1	AF_1	A^2F_1	A^3F_1	A^4	A^4	A^4	A^4
Shift-4 & Mult:	F_1	AF_1	A^2F_1	A^3F_1	A^4F_1	A^5F_1	A^6F_1	A^7F_1
RESULT:	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8

Is this algorithm cost-optimal? Explain your reasoning?

No, it is not cost optimal.
Because,
COST$_{seq}$ **=** **T**$_{seq}$ **= O(N)**
COST$_{par}$ **= P*T**$_{par}$ **= O(N*log$N$)**
COST$_{par}$ $>$ **COST**$_{seq}$

16. Describe an $O(logN)$ parallel hypercube algorithm to compute all the elements of the sequence $x_1,\ x_2,\ldots,\ x_N$ which are generated by the following recurrence relation:

$$x_1 = a_1, x_0 = b_1 \quad and \quad x_i = a_i.x_{i-1} + b_i.x_{i-2} + c_i \quad i = 2, 3, \ldots, N$$

Assume that, N=2^n processors are available and, initially, processor i holds a_{i+1}, b_{i+1}, and c_{i+1}.
Important Note: Your algorithm will compute, not only x_N, but also all the elements of the sequence $x_1,\ x_2,\ldots,\ x_N$. It is acceptable if you describe your algorithm in terms of the fundamental operations (i.e. broadcast, shift, sort, etc.) discussed in class without explaining them in detail.

$$X_1 = \begin{pmatrix} a_1 \\ b_1 \\ 1 \end{pmatrix} \quad and \quad X_i = \begin{pmatrix} x_i \\ x_{i-1} \\ 1 \end{pmatrix} = \begin{bmatrix} a_i & b_i & c_i \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_{i-1} \\ x_{i-2} \\ 1 \end{pmatrix} = A_i X_{i-1} \quad \forall i = 2, 3, \ldots N$$

```
 PE#:      0      1      2      3      4      5      6      7
          000    001    011    010    110    111    101    100   |   TIME
                                                                 | COMPLEXITY
--------------------------------------------------------------------------
```

Init State	X_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8

Step 1: **Processors execute Parallel Prefix Multiply** **O(log N)**

Shift-1 & Mult:
Shift-2 & Mult:
Shift-4 & Mult:

RESULT:	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8

17. Processor #0 wants to broadcast an item to all the other processors on a 16 processor hypercube using at most $O(logP)$ steps and nearest-neighbor communications. Draw the communication diagram for this broadcast algorithm (broadcast tree) which starts at processor #0.

```
   P0 (0000)
   |        \
   |            \
   |                \
   |                    \
   |                        \
   |                            \
   P0 (0000)                        P8 (1000)
   |       \                            |      \
   |           \                        |          \
   |               \                    |              \
   |                   \                |                  \
 0000                0100             1000               1100
   |  \                |  \             |  \               |  \
   |    \              |    \           |    \             |    \
   |      \            |      \         |      \           |      \
   |        \          |        \       |        \         |        \
 0000      0010      0100      0110   1000      1010     1100       1110
   |\        |\        |\        |\     |\        |\       |\         |\
   | \       | \       | \       | \    | \       | \      | \        | \
   |  \      |  \      |  \      |  \   |  \      |  \     |  \       |  \
   |   \     |   \     |   \     |   \  |   \     |   \    |   \      |   \
 0000 0001 0010 0011 0100 0101 ..   .. 1000 1001 1010 1011 1100 1101  1110 1111
```

18. (a) Explain and show all of the fundamental steps involved in computing the following polynomial of degree $(p-1)$ on a hypercube of p processors in O(log p) time.

$$S = \sum_{i=0}^{p-1} a_i.x^i = a_0x^0 + a_1x^1 + a_2x^2 + \ldots + a_{p-1}x^{p-1}$$

Assume that initially x value resides in p_0 and a_i is stored in the local memory of p_i, for $i = 0, 1, \ldots, p-1$. Draw a figure to show all of the computation and communication steps for p=8 and explain the time complexity for each step.

```
 PE#:        0      1      2      3      4      5      6      7
            000    001    011    010    110    111    101    100   |   TIME
                                                                   | COMPLEXITY
INITIAL      a0     a1     a2     a3     a4     a5     a6     a7   |
 STATE       x                                                     |
------------------------------------------------------------------------------
```

Step 1:	**P0 broadcasts x to all other processors (one-to-all broadcast)**								**O(log p)**
	x	x	x	x	x	x	x	x	
Step 2:	**Processors execute Parallel Prefix Multiply**								**O(log p)**
Shift-1 & Mult:									
	1	x	x^2	x^2	x^2	x^2	x^2	x^2	
Shift-2 & Mult:									
	1	x	x^2	x^3	x^4	x^4	x^4	x^4	
Shift-4 & Mult:									
	1	x	x^2	x^3	x^4	x^5	x^6	x^7	
Step 3:	**Processors multiply their results with the coefficients (a_i)**								**O(1)**
	$a_0.1$	$a_1.x^1$	$a_2.x^2$	$a_3.x^3$	$a_4.x^4$	$a_5.x^5$	$a_6.x^6$	$a_7.x^7$	

Step 4:	**Processors execute parallel reduction with OP="add"** **P0 gets** $S = a_0x^0 + a_1x^1 + a_2x^2 + \ldots + a_{p-1}x^{p-1}$	**O(log p)**

(b) Is this algorithm cost-optimal? Explain your reasoning?

No, it is not cost optimal.
Because,
COST$_{seq}$ **=** **T**$_{seq}$ **= O(p) (Using Horner's Rule)**
COST$_{par}$ **= p*T**$_{par}$ **= O(p*log**p**)**
COST$_{par}$ $>$ **COST**$_{seq}$

(c) If all a_i values initially reside in p_0 (i.e. they are not yet distributed to the local memories of the processors), how would your algorithm change and what would be the time complexity for the new algorithm? Is the new algorithm cost optimal?

We need an additional step to perform one-to-all scatter in order to distribute a_i **values to respective processors which would take O(**p**) time. Therefore,**
T$_{par}$ **= O(**p**)**
and It is still not cost-optimal.

(Problem Solving): END OF TEST I

---

1. (a) Show all the steps required (in the order shown below) to sort the following sequence in increasing order using *bitonic sort* algorithm on a 3-dimensional hypercube (please note that the number of steps given below may not be exact)

```
             P0   P1   P2  P3   P4   P5   P6   P7
             4    7    3    9    8    6    2    5
-------------------------------------------------------
STEP 1:      4    7    9    3    6    8    5    2
-------------------------------------------------------
STEP 2:      4    3    9    7    6    8    5    2
-------------------------------------------------------
STEP 3:      3    4    7    9    8    6    5    2
-------------------------------------------------------
STEP 4:      3    4    5    2    8    6    7    9
-------------------------------------------------------
STEP 5:      3    2    5    4    7    6    8    9
-------------------------------------------------------
STEP 6:      2    3    4    5    6    7    8    9
-------------------------------------------------------
```

(b) What is the time complexity of this algorithm for P=N?
T$_{par}$ **= O(log**^{2}**N)**

(c) Is it cost-optimal ? Explain your reasoning.

No. Because,

$\mathbf{T}_{seq} = \mathbf{O(NlogN)}$ ====> $\mathbf{COST}_{seq} = \mathbf{NlogN}$

$\mathbf{T}_{par} = \mathbf{O(log^2N)}$ ====> $\mathbf{COST}_{par} = \mathbf{Nlog^2N}$

$\mathbf{COST}_{seq} \neq \mathbf{COST}_{par}$

2. Given the following adjacency matrix for an undirected graph and a hypercube of 8 processors, we want to compute the Minimum Spanning Tree using Prim's algorithm.

```
          1  2  3  4  5  6  7  8
        _                        _
     1 |  0  2  5  -  -  -  4  1  |
     2 |  2  0  -  3  -  -  -  4  |
     3 |  5  -  0  2  -  -  6  -  |
 A = 4 |  -  3  2  0  1  -  -  -  |
     5 |  -  -  -  1  0  2  -  3  |
     6 |  -  -  -  -  2  0  -  -  |
     7 |  4  -  6  -  -  -  0  -  |
     8 |  1  4  -  -  3  -  -  0  |
        -                        -
```

(a) (X points) Start at vertex #1 and write the contents of D vector after each step and indicate the vertex# selected at each step (in case of a tie, pick the smaller vertex no.):

	V:	1	2	3	4	5	6	7	8	Vertex# Selected
INITIAL :	D:	0	2	5	-	-	-	4	1	8
After Step 1:	D:	0	2	5	-	3	-	4	1	2
After Step 2:	D:	0	2	5	3	3	-	4	1	4
After Step 3:	D:	0	2	2	3	1	-	4	1	5
After Step 4:	D:	0	2	2	3	1	2	4	1	3

(b) (X points) What is the sequential time complexity of Prim's MST algorithm for an undirected graph with N vertices and E edges. Assume that the graph is dense (i.e. E $\approx$ N^2)

$\mathbf{T}_{seq} = \mathbf{O(N^2)}$

(c) (X points) What is the time complexity for the most efficient parallel implementation of Prim's MST algorithm on P=N processors.

$\mathbf{T}_{par} = \mathbf{N*(N/P + logP) = N + NlogN = O(NlogN)}$

(d) (X points) Calculate the *speedup* and *efficiency* for your parallel algorithm.

Speedup $= \frac{T_{seq}}{T_{par}} = \frac{N^2}{NlogN} = \frac{N}{logN}$

Efficiency $= \frac{Speedup}{P} = \frac{N/logN}{N} = \frac{1}{logN}$

3. **(15 pts)** Consider Cannon's matrix multiplication algorithm described in class where $P = N^2$ processors were used. Here, you are asked to design an algorithm to multiply two matrices A and B, each of size N^2, on a hypercube of P^2 processors (where $N = k * P$ and $k \geq 1$) in $O(k^3P)$ time using at most $3k^2$ memory per processor.

(a) Explain the steps of your algorithm clearly by drawing a figure for $P^2 = 16$ and $N^2 = 8 * 8 = 64$.

(b) Would you agree that this is *the optimum* (with respect to time and space complexity) parallel matrix multiplication algorithm possible on a multicomputer? Explain your reasoning clearly referring to time and space complexities and why they are optimum.

Use (Cannon's Algorithm) + (Block Matrix Multiplication).
Dimension of each block matrix is $k*k$ where k = N/P. Show the initial steps of the Cannon's algorithm and explain how the initial alignment is done. Also show that there will be a total of O(P) steps each taking O(k^3) time. Therefore, the total parallel time will be O(N^3/P^2) which indicates *linear speedup* and hence, cost optimality. As far as the space complexity is concerned, you should argue that any matrix multiplication algorithm need to use at least $3N^2$ storage space and your algorithm is using exactly this much total storage, and furthermore, the storage requirements is evenly divided among the processors (i.e. each PE needs $3N^2/P^2 = 3k^2$ storage).

4. Given the following adjacency matrix for an undirected graph and a hypercube of 8 processors, we want to compute the Single Source Shortest Paths to vertex #1 using Dijsktra's algorithm.

```
           1  2  3  4  5  6  7  8
        -                          -
    1 |    0  2  6  -  -  -  5  1  |
    2 |    2  0  -  1  -  -  -  4  |
    3 |    6  -  0  2  -  -  2  -  |
A = 4 |    -  1  2  0  1  -  -  -  |
    5 |    -  -  -  1  0  2  -  -  |
    6 |    -  -  -  -  2  0  -  -  |
    7 |    5  -  2  -  -  -  0  -  |
    8 |    1  4  -  -  -  -  -  0  |
        -                          -
```

(a) (X points) Start at vertex #1 and write the contents of L vector after each step and indicate the vertex# selected at each step (in case of a tie, pick the smaller vertex no.):

	V:	1	2	3	4	5	6	7	8	Vertex# Selected
INITIAL :	L:	0	2	6	-	-	-	5	1	
After Step 1:	L:	0								
After Step 2:	L:	0								
After Step 3:	L:	0								
After Step 4:	L:	0								

ANSWER:

	V:	1	2	3	4	5	6	7	8	Vertex# Selected
INITIAL :	L:	0	2	6	-	-	-	5	1	8
After Step 1:	L:	0	2	6	-	-	-	5	1	2
After Step 2:	L:	0	2	6	3	-	-	5	1	4
After Step 3:	L:	0	2	5	3	4	-	5	1	5
After Step 4:	L:	0	2	5	3	4	6	5	1	3

(b) (X points) What is the sequential time complexity of Dijkstra's Single-Source Shortest Paths (SSSP) algorithm for an undirected graph with N vertices and E edges. Assume that the graph is dense (i.e. E ≈ N^2)

$\mathbf{T}_{seq} = \mathbf{O(N^2)}$

(c) (X points) What is the time complexity for the most efficient parallel implementation of Dijkstra's SSSP algorithm on P=N processors.

$\mathbf{T}_{par} = \mathbf{N^*(N/P + logP) = N + NlogN = O(NlogN)}$

(d) (X points) Calculate the *speedup* and *efficiency* for your parallel algorithm and comment about cost optimality.

Speedup $= \frac{T_{seq}}{T_{par}} = \frac{N^2}{NlogN} = \frac{N}{logN}$

Efficiency $= \frac{Speedup}{P} = \frac{N/logN}{N} = \frac{1}{logN}$

It is NOT cost-optimal, because $COST_{seq}$ $=$ **O(N**^{2}**)** $<$ $COST_{par}$ $=$ **N*NlogN**

(Problem Solving): END OF TEST II

(Problem Solving): ADDITIONAL QUESTIONS

---

1. Prove that average distance between any two processors on a hypercube is $\frac{d}{2}$ where d is the dimension of the hypercube.

2. Processor #6 wants to broadcast an item to all the other processors on a 16 processor hypercube using at most $O(logP)$ steps and nearest-neighbor communications. Draw the communication diagram for this broadcast algorithm (broadcast tree) which starts at processor #6. (Hint: If we bitwise-XOR both X and Y with a number N to obtain $\hat{X} = X \oplus N$ and $\hat{Y} = Y \oplus N$, the Hamming Distance between $\hat{X}$ and $\hat{Y}$ will be the same as that of X and Y)

3. In a bus-oriented shared memory multiprocessor system, there are P processors connected to the bus and they have no local memory. Assume that, with this current configuration, P processors use up the total (bus and shared memory) bandwidth. If each processor is provided with a local cache memory, by what factor can P (the number of PEs) be increased before the shared memory becomes a bottleneck. Assume that the shared memory bandwidth is B (bytes per second), and the *hit ratio* for a local cache is the same for all caches and is equal to h. Show your work.

4. Given the following mask configuration for edge detection.

w_0	w_1	w_2
w_3	w_4	w_5
w_6	w_7	w_8

(a) If we decide to use the *Sobel Operator*, show the masks for calculating $\delta f/\delta x$ and $\delta f/\delta y$. And write their expressions in terms of image pixel values.

(b) Write the expressions for *Gradient Magnitude* ($\bigtriangledown f$) and *Gradient Direction* ($\Phi(x, y)$).