Q.1. State and explain the different criteria used in evaluating networks of computers for parallel computing. For each criterion, state whether a high value or a low value is desirable, and explain the reason for your answer. Compare the \( d \)-dimensional hypercube network, and the \( d \)-dimensional cube-connected cycles network for each of the above criteria by stating the values for each criterion. (10 points)

A.1. The criteria used in evaluating networks of computers for parallel computing are:

**Diameter** The diameter of a network is the maximum of the minimum distances between every pair of nodes in the network. A low diameter is preferable because the diameter places a lower bound on the communication time between nodes in the network.

**Bisection width** The bisection width of a network is the minimum number of edges that must be removed in order to divide the network into roughly two equal halves. A high bisection width is preferable as this bisection width puts a lower bound on the amount of data that can be transferred between two parts of the network.

**Degree** The degree of a network is the maximum number of edges per node. For scalability of the network, it is preferable to have the degree be a constant, independent of the size of the network.

**Maximum edge length** This is the maximum, physical length of the connections between nodes when the network is laid-out in 3-dimensional space. For scalability, and speed of communication, it is preferable if the maximum edge length is a constant, independent of the size of the network.

Below are the values for the above criteria for the \( d \)-dimensional hypercube network, and the \( d \)-dimensional cube-connected cycles network.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>( d )-cube</th>
<th>( d )-CCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diameter</td>
<td>( \Theta(d) )</td>
<td>( \Theta(d) )</td>
</tr>
<tr>
<td>Bisection width</td>
<td>( 2^{k-1} )</td>
<td>( 2^{k-1} )</td>
</tr>
<tr>
<td>Degree</td>
<td>( d )</td>
<td>3</td>
</tr>
<tr>
<td>Edge Length</td>
<td>Not constant</td>
<td>Not constant</td>
</tr>
</tbody>
</table>
Q.2. Explain what is a cost-optimal parallel algorithm. (5 points)

A.2. Suppose a given parallel algorithm for a problem uses $p$ processors, and has a running time of $T(n)$ on an input of size $n$. Then, the cost of the parallel algorithm is defined to be $p \cdot T(n)$. Suppose the optimal sequential algorithm for the same problem has a running time of $S(n)$ on an input of size $n$. Then, the parallel algorithm is called cost-optimal if and only if

$$p \cdot T(n) \leq S(n).$$

Q.3. Describe a cost-optimal parallel algorithm, with running time $O(\log n)$, for computing the sum of $n$ integers. Your description should (1) clearly state how many processes are used, (2) contain a precise description of the algorithm, (3) contain a precise derivation of the running time of the algorithm, and (4) show that the algorithm is cost-optimal. (15 points)

A.3. This algorithm uses $p = \frac{n}{\log n}$ processors and assumes that the array of $n$ elements is in global memory. The processors are labelled $P_i$, $0 \leq i < p$.

**Step 1** For each $i$, $0 \leq i < p$, $P_i$ computes the sum of the $\log n$ elements starting at the index $(i - 1) \log n$, and stores this sum in $s_i$.

**Step 2**

For each $j$, $0 \leq j \leq \log p$, for each $i$, $0 \leq i < p$, if $i = 2^{j+1} \cdot k$, for some $k \geq 0$,

$$s_i \leftarrow s_i + s_{i+2^j}.$$ 

In the last statement above, if $i + 2^j \geq p$, then the value of $s_{i+2^j}$ is assumed to be zero.

Since each processor adds up $\log n$ elements, Step 1 takes $O(\log n)$ time using a sequential algorithm. Step 2 takes $O(\log p)$ time, as each time half the processors become “inactive”, eventually leaving $P_i$ with the sum of all the elements. Thus, Step 2 takes $O(\log n - \log \log n)$ time. Thus the entire algorithm takes $O(\log n)$ time.

The cost of the above algorithm is

$$p \cdot O(\log n) = \frac{n}{\log n} \cdot O(\log n) = O(n).$$

Thus the algorithm is cost optimal.

Q.4. State and explain the expression for computing the speedup of a parallel algorithm. (5 points)

A.4. The speedup of a parallel algorithm for a given problem is defined to be the ratio between the running time of the most efficient sequential algorithm for the problem and the running time of the given parallel algorithm.

Q.5. State Amdahl’s law, and give a brief explanation of what the law means. (5 points)

A.5. Amdahl’s law gives an upper bound on the speedup, $S$, that can be achieved by a given parallel algorithm that solves a given problem. Suppose $f$ is the fraction of operations that must be performed sequentially to solve the given problem. Suppose the given parallel algorithm uses $p$ processors. Then,

$$S \leq \frac{1}{f + (1 - f)/p}.$$
Q.6. Describe the differences between the RAM and the PRAM models of computation. Explain the various conventions for the PRAM model. Which version is computationally most powerful? Explain your answer. (5 points)

A.6. In the RAM model of computation there is one processor (finite state control mechanism), unbounded local memory, and unbounded global memory.

In the PRAM model there are an unbounded number of identical processors, each with its own unbounded local memory, and unbounded global memory. All the processors execute the same instruction synchronously.

Algorithms written for the PRAM model can be classified as:

- **EREW** Exclusive Read Exclusive Write. In such algorithms, no two processors access (for reading or writing) the same global memory location at the same time.
- **CREW** Concurrent Read Exclusive Write. In such algorithms, no two processors write to the same global memory location at the same time, but two processors may concurrently read from the same global location.
- **ERCW** Exclusive Read Concurrent Write. In such algorithms, no two processors read from the same global memory location at the same time, but two processors may concurrently write from the same global location.
- **CRCW** Concurrent Read Concurrent Write. In such algorithms, two processors may concurrently access the same global memory location at the same time for both, reading or writing.

In the case of concurrent write algorithms, the PRAM model must follow some convention to decide which of the several processors writing to the same global memory location “wins”. One of the following conventions is usually adopted. (By competing processors, we mean processors that are concurrently writing to the same location in global memory.)

- **COMMON** All competing processors must write the same value.
- **ARBITRARY** One of the competing processors is arbitrarily chosen as the “winner”.
- **PRIORITY** The competing processor with the lowest index (highest priority) is chosen as the “winner”.

The CRCW model with the PRIORITY convention is computationally the most powerful since algorithms written for any other model can be executed on this model with exactly identical behaviour and running time.

Q.7. Prove that in a $d$-dimensional wrapped butterfly, if the straight edges are removed, then the remaining edges form $2^{d-1}$ disjoint cycles. (15 points)

A.7. Suppose $G$ is the graph of the $d$-dimensional wrapped butterfly where the straight edges are removed. We will refer to vertices in rows and levels of $G$ similar to nodes in the $d$-dimensional wrapped butterfly.

**Claim 1** All vertices of $G$ belong to some cycle in $G$.

**Proof** In the $d$-dimensional wrapped butterfly, each vertex has degree 4. At each vertex, two of these edges are the straight edges. Thus, in $G$, each vertex has degree 2, and thus must be in some cycle. Claim 1
Claim 2 Suppose $V(l)$ is the set of vertices at level $l$ in $G$. For each $v$ in $V(l)$, there is a unique vertex $v'$ in $V(l)$ such that $v'$ is connected to $v$.

Proof Suppose $v$ is in $V(l)$. Since all the straight edges are removed from the wrapped butterfly, $G$ contains only the cross edges of the wrapped butterfly. Consider the neighbours of vertices along the forward cross edges. Let $C(v) = v_0, v_1, \ldots, v_k$, where $v = v_0 = v_k$ be the cycle containing $v$, such that for each $i$, $v_{i+1}$ is the neighbour of $v_i$ along the forward cross edge. For any vertex $u$, let $L(u)$ denote the level of $u$. By the definition of the wrapped butterfly, $L(v_{i+1}) = (L(v_i) + 1) \mod d$, and $v_i$ and $v_{i+1}$ differ in bit position $L(v_{i+1})$. Therefore, $L(v_d) = L(v_0)$, and $v_d$ and $v_0$ differ in all bit positions. A similar argument shows that $k = 2d$, establishing that, in $C(v)$, only the vertex $v_d$ is at the same level as $v$.

Claim 1 above guarantees that $G$ is made up of cycles. Claim 2 guarantees that of all the vertices at each level, exactly two are in one cycle. Since each level of $G$ has $2^d$ vertices, there must be $2^{d-1}$ disjoint cycles in $G$. 

\[ \square \text{ Claim 2} \]