**Assignment 2**

Umamah Hussain

21L-1858

**Q.1**

**Suppose an array of size N is to be mapped on P processes numbered as [ p0, p1, ......, pP-1] using array-block-distribution scheme, then answer the following questions:**

1. **How many tasks/blocks will be there?**

Number of tasks/blocks = Number of processes

1. **What will be the size of each block?**

The size of each block will be determined by dividing the total size of the array (N) by the number of processes (P).

**Block Size = N / P**

**c) If block number starts from zero then, to which process number the i(th) block will be assigned?**

To determine which process number the i(th) block will be assigned to, we can use the formula **Process Number = i % P**

Where

i= Block number

P = Total Processes

1. **Suppose array indexing starts from zero, what will be the starting index of the block assigned to i(th) process?**

**Start index = i \* Block Size**

Where

i = process number

1. **Suppose array indexing starts from zero, what will be the end index of the block assigned to i(th) process?**

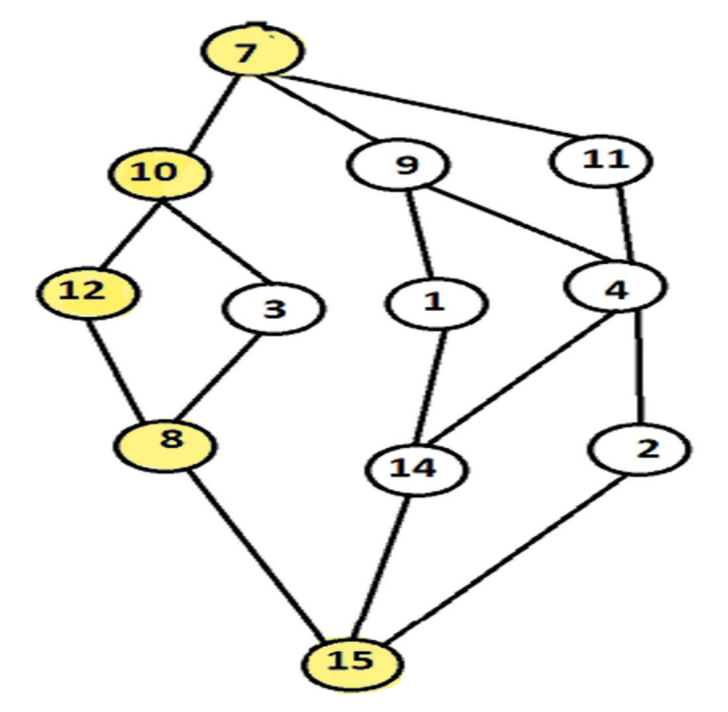
**Start index = (i+1) \* Block Size**

Where

i = process number

**Q.3**

1. **Critical path**

****

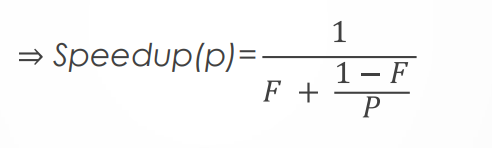
**Critical Path= 7 -> 10 -> 12 -> 8 -> 15**

1. **Critical path length**

**Critical Path= 7 + 10 + 12 + 8 + 15**

**Critical Path= 52**

1. **Maximum possible speedup assuming large number of processes are available**

****

Where

**F= Serial portion of the code**

**1-F= Parallelizable portion of the code**

**P= Number of processors**

Amdahl's Law emphasizes that the speedup achievable by parallelizing a computation is limited by the fraction of the computation that must be executed serially. Even if you parallelize a portion of the computation perfectly, the presence of serial portions will limit the overall speedup. The serial portion remains constant, constraining the overall speedup that can be achieved.

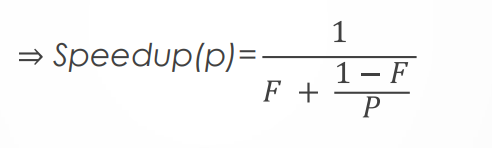
When a large number of processes are available, the number of processors p becomes large. As p increases, the term 1-F/p​ becomes smaller, approaching zero.

In this scenario, the dominant term in the denominator becomes F, as the term 1-F/p​ ​ becomes negligible compared to F. Thus, the denominator becomes approximately F.

Therefore, the maximum possible speedup S tends towards 1/F as the number of processors P becomes large, when a large number of processes are available.

1. **Minimum number of processes needed to obtain the maximum possible speedup**

To obtain the maximum possible speedup, we want to minimize the denominator in the expression for speedup:

****

Where:

**F is the fraction of the computation that must be executed serially.**

**1−F is the fraction of the computation that can be parallelized.**

**p is the number of processors.**

To minimize the denominator, we want to maximize the value of p. The minimum number of processes needed to obtain the maximum possible speedup is when p is maximized, which means using as many processors as possible.

Therefore, the minimum number of processes needed to obtain the maximum possible speedup is when p is equal to the maximum available number of processors.

1. **Maximum speed up if number of processes are limited to 4**

Given that the number of processes is limited to 4, p=4.

We also need to know the values of F and 1−F, the fractions of the computation that must be executed serially and can be parallelized, respectively.

Once we have those values, we can plug them into the formula to calculate the maximum speed up.