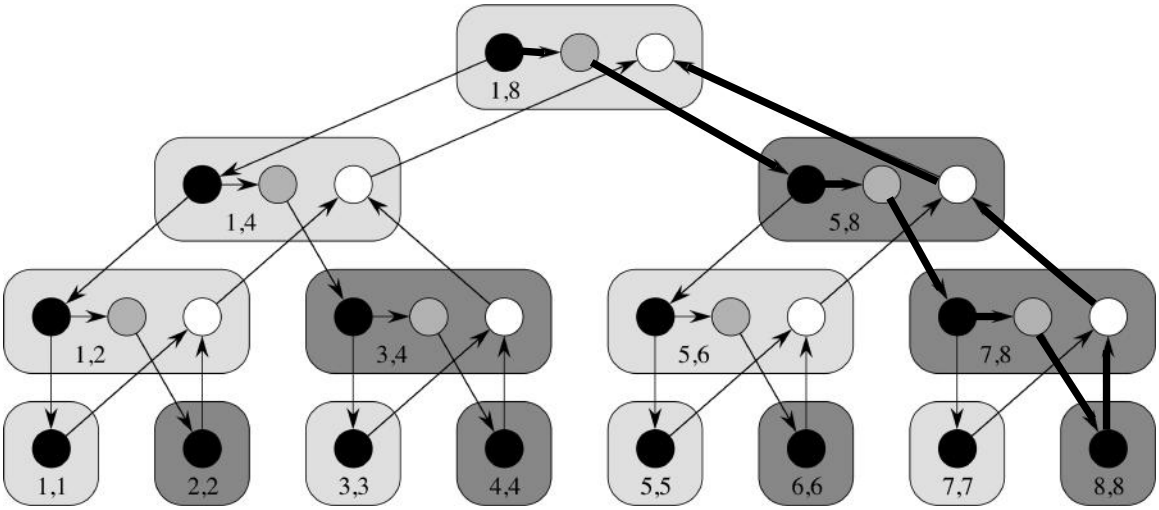
Name: _____

1. (4 points) The algorithm P-SUM(L, p, r) computes the sum of the elements $L[p\,..\,r]$ of an array L of length n. Draw the DAG for P-SUM$(L, 1, 8)$ where L is an array of length 8. Determine the work, span, and parallelism from the DAG. Show all work.

```
P-SUM(L, p, r)
1  if p < r
2       q = ⌊(p + r)/2⌋
3       x = spawn P-SUM(L, p, q)
4       y = P-SUM(L, q + 1, r)
5       sync
6       return x + y
7  else
8       return L[p]
```

Solution



The critical path is bolded. The span is the length of the critical path (in strands, not edges), so $T_\infty = 10$. The work is the number of strands, so $T_1 = 29$.

The parallelism is $T_1/T_\infty = 2.9$.

(continued on other side)

2. (4 points) The algorithm P-SEARCH(L, p, r, v) performs parallel linear search for the value v in the subarray $L[p\,..\,r]$ of an array L of length n. It returns TRUE if v is found in $L[p\,..\,r]$ and returns FALSE otherwise.

```
P-SEARCH(L, p, r, v)
1  if p < r
2       q = ⌊(p + q)/2⌋
3       x = spawn P-SEARCH(L, p, q, v)
4       y = P-SEARCH(L, q + 1, r, v)
5       sync
6       return (x || y) // Logical 'or' of x and y
7  else
8       return (L[p] == v) // TRUE if L[p] has the value v; FALSE otherwise
```

Determine the work, span, and parallelism of P-SEARCH$(L, 1, n, v)$ for an n-long array L. What is the *parallel slackness* when $n = 256$ and $P = 16$? Justify your answers.

Solution

The work is just the serial running time of the algorithm. Let $n = r - p + 1$. The function makes two recursive calls of size $n/2$, and all other work is $\Theta(1)$, so $T_1(n) = 2T_1(n/2) + \Theta(1) = \Theta(n)$ (by the Master Theorem or an easy argument using the recursion tree). The computation of the span is similar, except the two recursive calls will execute in parallel, giving $T_\infty(n) = T_\infty(n/2) + \Theta(1) = \Theta(\lg n)$.

The parallelism is $T_1(n)/T_\infty(n) = \Theta(n/\lg n)$ and the parallel slackness when $n = 256$ and $P = 16$ is

$$\frac{T_1(n)}{P \cdot T_\infty(n)} = \frac{256}{16 \cdot 8} = 2.$$

3. (2 points) The function P-Square-Matrix-Mul(A, B) computes the product of two n-by-n matrices A and B. The implementation includes a *race condition*. Locate and describe the race condition and indicate how the code should be modified to fix the problem.

```
P-Square-Matrix-Mul(A, B)
1  n = A.rows
2  let C be a new n-by-n matrix
3  parallel for i = 1 to n
4       parallel for new j = 1 to n
5            C_ij = 0
6            for k = 1 to n
7                 C_ij = C_ij + A_ik · B_kj
8  return C
```

Solution

The problem is the lack of "**new** k" in the loop on line 6. The parallel for loop on line 4 may create a separate thread for each value of j, with each thread executing lines 6 – 7 *with a common k variable*. Adding the **new** keyword will fix the problem.