❄ 332 Final Prep Answers ❄

1. Concurrency

Annie and Ollin share a jug of water. Rather than going to the store and buying a new bottle whenever they run out (so wasteful!), they continually refill the same jug. In addition, they keep track of how many cups are left in the jug on a sticky note so that they don't have to look in the jug before adding water to it or pouring themselves a delicious glass of water.

```
private int cupsOfWater;
private int maximumWater;
private Stack<Water> water;
private int checkWater() {
    return cupsOfWater;
}
private Water pourWater() {
    Water e = water.pop()
    cupsOfWater--;
    return e;
}
public Water getWater() {
    if (checkWater() > 0) {
        return pourWater();
    }
    // cryDeeply();
}
public void addWater (Water e) {
    if (cupsOfWater =/= maximumWater) {
        water.push(e);
        cupsOfWater++;
    }
}
```

1.1 Let's say that each TA is a thread. Provide an interleaving of the two TAs that causes the maximum amount of embarassment.

```
if (cupsOfWater =/= maximumWater)
                                        if (cupsOfWater =/= maximumWater)
water.push(e)
                                        water.push(e)
```

1.2 Let's say we synchronize `cupsOfWater` and `water`. Does this make our scenario thread-safe?

Nope, you can still have bad interleavings like the one above.

1.3 Due to budget cuts we also share one singular cup now, and, having learned our lesson from last time, we decide to synchronize access to this cup using a new lock. What problems could this cause?

Deadlock (if we synchronize both the cup and the jug)

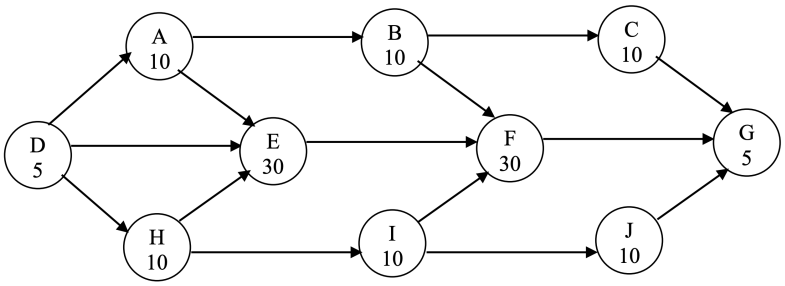
2. Parallelism Theory

2.1 Given a program where 75% of it is parallelizable (and 25% of it must be run sequentially) what is the maximum speedup you would expect to get with 5 processors? Note: You must show your work for any credit. For full credit give your answer as a number or a simplified fraction (not a formula).

$$\text{Speedup on } P \text{ processors} = \frac{T_1}{T_P} = \frac{1}{S + \left(\frac{1-S}{P}\right)} = \frac{1}{0.25 + \left(\frac{0.75}{5}\right)} = \frac{1}{25 + 0.15} = \frac{1}{0.40} = \frac{10}{4} = \frac{5}{2} = 2.5$$

2.2 Consider the following directed acyclic graph representing the dependencies in a parallel computation implemented using a fork / join technique.

Each vertex is annotated with the cost of performing its work.



a) What is the work of this computation (i.e., a number)?

130

b) More generally, what is the work of a computation represented in this manner (i.e., described in terms of the graph and the cost of each vertex)?

Sum of the work in all vertices.

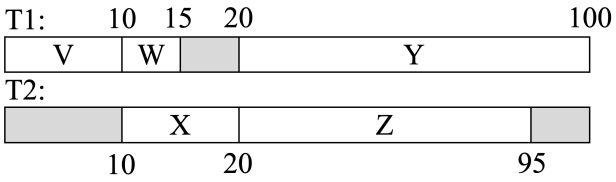
c) What is the span of this computation (i.e., a number)?

80

d) More generally, what is the span of a computation represented in this manner (i.e., described in terms of the graph and the cost of each vertex)?

Sum of the work in the most expensive path in the graph.

e) Assume **two threads** are executing a computation. We can illustrate the parallel work of multiple threads by drawing timelines of the work they execute. For example:

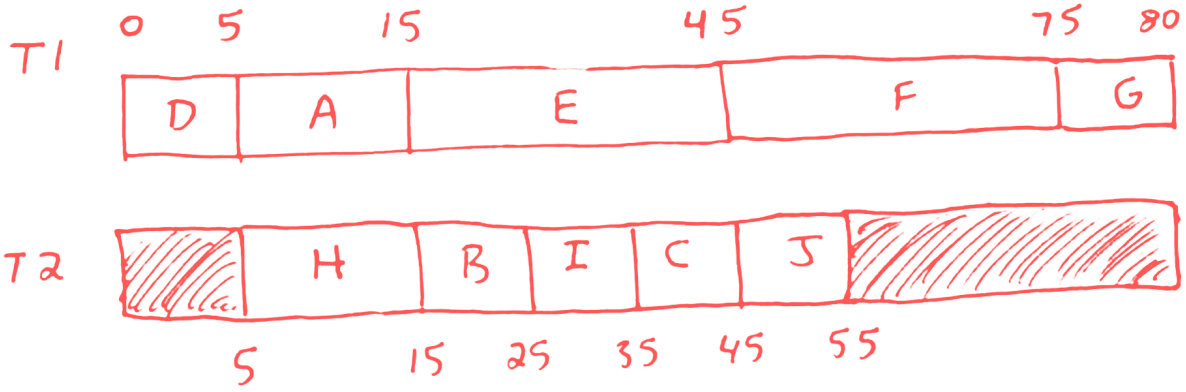


This pair of timelines illustrates a hypothetical computation in which:

T1: V for 10 units, W for 5 units, idle for 5 units, Y for 80 units

T2: idle for 10 units, X for 10 units, Z for 75 units, idle for 5 units

Draw a timeline using **two threads** to execute the graph from the previous page as quickly as possible. Be sure your timeline illustrates the start and stop time of each task on each thread.



f) Would additional threads be able to perform the computation more quickly? Why?

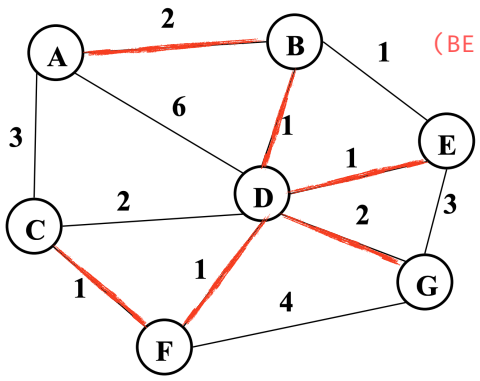
No. The timeline completes in 80, which is the span.

3. Graphs!

3.1 What is the big-O running time of Prim's algorithm (assuming an adjacency list representation) if a priority queue is used?

O(E log V)

3.2 Give **a** Minimum Spanning Tree (MST) of the graph below, by highlighting the edges that would be part of the MST.



(BE could be swapped for BD or DE)

3.3 Kruskal's

a) What is the worst case running time of Kruskal's algorithm as described in lecture (assuming an adjacency list representation is used)?

O(E log E) or O(E log V)

b) You try using a new implementation of union-find that claims to have better data locality. `find()` in this new implementation has a worst case running time of $O(V^2)$ and `union()` has a running time of O(V). What is the worst case running time of your modified Kruskal's algorithm that uses this new implementation of union-find?

O(E log E + 2E * V^2 + V * V) or O(E * V^2)

3.4 What is the worst case running time to determine whether an edge exists from vertex x to vertex y...

a) Given an adjacency matrix representation:

O(1)

b) Given an adjacency list representation:

O(V) or O(d) where d is out-degree of vertex x