Basic Techniques.

This part will test your ability to apply techniques that have been explicitly identified in lecture and reinforced through sections and homeworks. Remember to show your work and justify your claims.

2. RCTOA NDECOIIN [10 points]

Imagine we run `main`. Is there a race condition? If there is one, explain why by showing a bad interleaving and explaining what the race is. If not, explain why not.

```
1  public static Lock lock = new ReentrantLock();
2  public static Stack<Integer> stack;

3  public static void main(String[] args) {
4     stack = <initialize stack with elements>;
5
6     Task t1 = <run task>;
7     Task t2 = <run task>;
8     t1.fork();
9     t2.fork();
10    int sum = t1.join() + t2.join();
11    System.out.println("sum = " + sum);
12 }
```

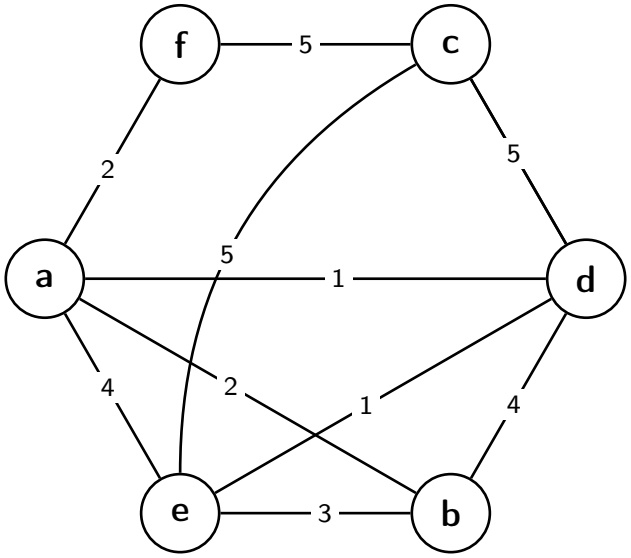
```
13 public int task() {
14    int count = 0;
15    while (!stack.isEmpty()) {
16       lock.lock();
17       count += stack.pop();
18       lock.unlock();
19    }
20    return count;
21 }
```

Solution:

`int count = 0`	
	`int count = 0`
`while (!stack.isEmpty())`	
	`while (!stack.isEmpty())`
`lock.lock()`	
`count += stack.pop()`	
`// stack is now empty!`	
`lock.unlock()`	
	`lock.lock()`
	`count += stack.pop()`
	`// stack is now empty!`
	`lock.unlock()`
`return count`	
	`return count`

Consider a stack containing exactly one element. When both stacks start, they both call `stack.isEmpty()`, both obtain `false`, and so both enter the while loop. Then, both tasks will attempt to pop something from the stack. One will succeed, but then the other will fail and throw an exception. An exception is the incorrect behavior, as we don't expect the program to crash.

4. Spring Time! [10 points]



(a) (5 points) Use Kruskal's Algorithm to find *two* minimum spanning trees of the above graph.

Solution: Three possible answers are:

- e-d, a-d, a-f, a-b, f-c
- e-d, a-d, a-f, a-b, c-d
- e-d, a-d, a-f, a-b, e-c

On an exam, we would expect you to show the forests, when each edge is chosen, etc. Just showing the answer would not get any credit.

(b) (5 points) Imagine that the above graph had some negative edges in it. Would Prim's Algorithm necessarily return a correct result? Explain your answer in 1-2 sentences.

Solution: Prim's algorithm **will** return the correct result, even with negative edges. Prim's algorithm works only by doing a relative comparison between the different edges currently under comparision and will pick the cheapest. This is in contrast to Dijkstra's, which both compares AND sums up the edges and assumes that each new edge increases the cost of the path.

5. Definitely A Graph! [10 points]

(a) (5 points) Suppose you are given a graph G. Explain how you would figure out if it has a cycle.

Solution: To find if a *directed* graph has a cycle, run topological sort. If the number of vertices at the end is not the number of total vertices, there is a cycle.

To find if an *undirected* graph has a cycle, run DFS. If you ever hit a node that you've already visited, then there is a cycle.

(b) (5 points) Suppose you are given a DAG G representing the work graph of a bunch of `ForkJoin` tasks. Explain how you would determine the longest dependency path of G. What does the longest dependency path in G represent with respect to the algorithm G?

Solution: Run DFS on every node with an in-degree of zero. Keeping track of the longest path found along the way.

Since this DAG represents ForkJoin operations, the longest dependency path represents the span of whatever algorithm is doing the forking and joining.

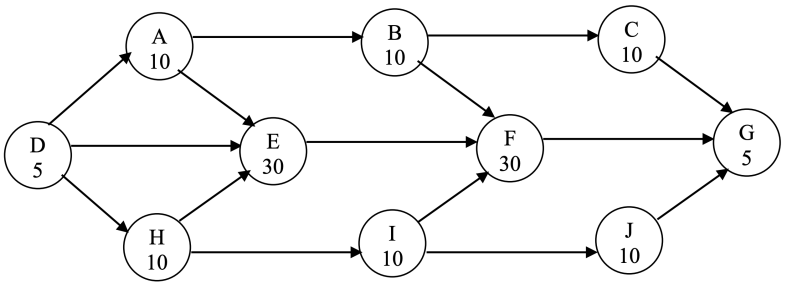
2. Parallelism Theory

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.

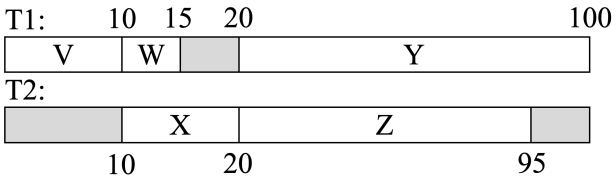
c) What is the span of this computation (i.e., a number)?

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:

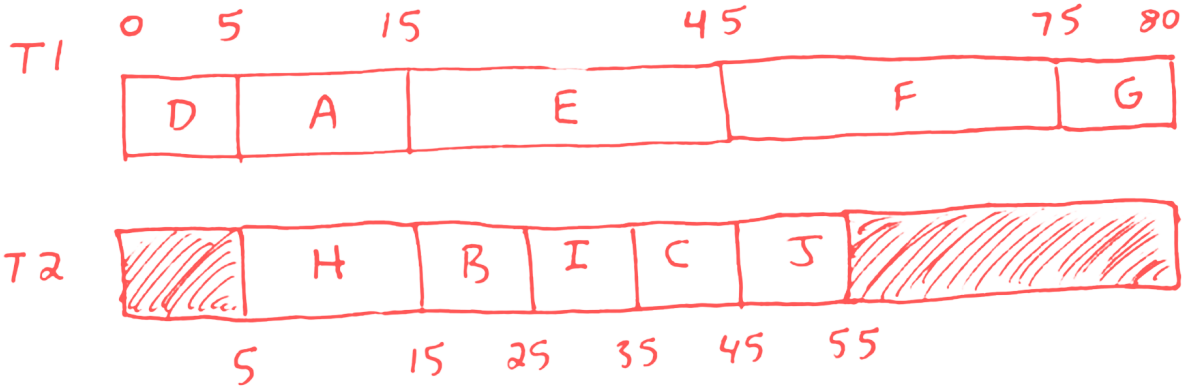


This pair of timelines illustrates a hypothetical computation in which:

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.

9) [6 points] Speedup

Your boss wants 111x speedup on a program of which 9/10 is parallelizable. What do you tell them? At least how many processors would you need?

Justify your answer with a computation. No credit given without an explanation.

$\frac{T_1}{T_\infty} = \frac{1}{S} = \frac{1}{1/10} = 10$ ← Maximum Speedup possible for a program with 1/10 that must be run sequentially.

Tell your boss that the max speedup possible on this program is 10x. So a 111x speedup will not be possible.