Warm-Up

We said last time bank account should have 1 lock per object, not 1 lock per method that updates the balance. Show a bad interleaving when we have a lock for each method.

```
class BankAccount{
 private int balance = 0;
 private Lock lk = new Lock();
 private Lock lk2 = new Lock();

void withdraw(int amount){
   lk.acquire(); //might block
   int b = getBalance();
   if(amount > b) {
     lk.release();
     throw new
  WithdrawTooLargeException();
   setBalance(b – amount);
   lk.release();
 }
```

```
void deposit(int amount){
    lk2.acquire();
   int b = getBalance();
   setBalance(b + amount);
   lk2.release();
 }
```

3

Huh?

Consider this code…

```
class Stack<E>{
  private E[] array = (E[])new Object[SIZE];
  private int index = -1;
  synchronized public Boolean isEmpty() { return index==-1;}
  synchronized public void push(E val) {array[++index]=val;}
  synchronized public E pop() {
    if(isEmpty()) { throw new StackEmptyException(); }
    return array[index--];
  }
  public E peek() { E ans = pop(); push(ans); return ans; }
}
```

21

Minimum Spanning Trees

What do we need? A set of edges such that:

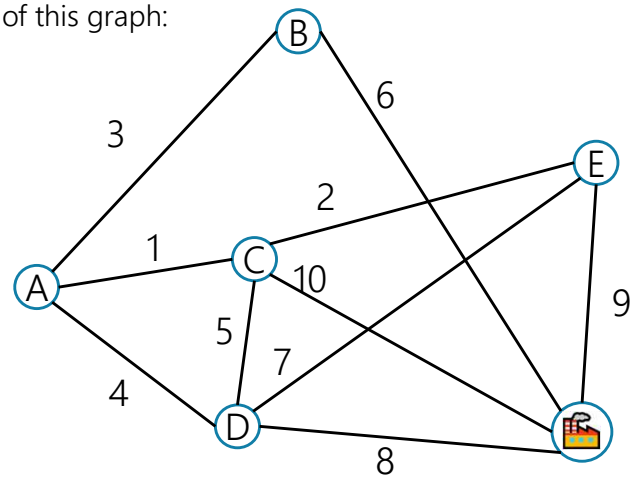
- Every vertex touches at least one of the edges. (the edges **span** the graph)
- The graph on just those edges is **connected**.
  - i.e. the edges are all in the same **connected component.**
  - A connected component is a vertex and everything you can reach from it.
- The minimum weight set of edges that meet those conditions

Claim: The set of edges we pick never has a cycle. Why?

38

Example

Try to find an MST of this graph:



41