```
from concurrent.futures import Future
from unittest.mock import Mock, create_autospec, patch

from swarms.structs import Agent, ConcurrentWorkflow, Task


def test_add():
    workflow = ConcurrentWorkflow(max_workers=2)
    task = Mock(spec=Task)
    workflow.add(task)
    assert task in workflow.tasks


def test_run():
    workflow = ConcurrentWorkflow(max_workers=2)
    task1 = create_autospec(Task)
    task2 = create_autospec(Task)
    workflow.add(task1)
    workflow.add(task2)

    with patch(
        "concurrent.futures.ThreadPoolExecutor"
    ) as mock_executor:
        future1 = Future()
        future1.set_result(None)
        future2 = Future()
```

```
        future2.set_result(None)

        mock_executor.return_value.__enter__.return_value.submit.side_effect = [
            future1,
            future2,
        ]
        mock_executor.return_value.__enter__.return_value.as_completed.return_value = [
            future1,
            future2,
        ]

        workflow.run()

    task1.execute.assert_called_once()
    task2.execute.assert_called_once()


def test_execute_task():
    workflow = ConcurrentWorkflow(max_workers=2)
    task = create_autospec(Task)
    workflow._execute_task(task)
    task.execute.assert_called_once()


def test_agent_execution():
    workflow = ConcurrentWorkflow(max_workers=2)
```

```
agent = create_autospec(Agent)

task = Task(agent)

workflow.add(task)

workflow._execute_task(task)

agent.execute.assert_called_once()
```