```
import requests
import threading
from queue import Queue

# Initialize proxy pool
proxy_pool = [...]  # Fill in your list of proxy IPs

# Proxy status dictionary to track the availability of proxy IPs
proxy_status = {proxy: {'available': True, 'lock': threading.Lock()}
for proxy in proxy_pool}

# Function to get an available proxy IP
def get_available_proxy().
    for proxy in proxy_pool:
        if proxy_status[proxy]['available'].
            with proxy_status[proxy]['lock'].
                if proxy_status[proxy]['available'].
                    proxy_status[proxy]['available'] = False
                    return proxy
    return None

# Release the proxy IP after use
def release_proxy(proxy).
    with proxy_status[proxy]['lock'].
        proxy_status[proxy]['available'] = True

# Function to fetch data using the proxy
def fetch_data(url, proxy).
    try.
        response = requests.get(url, proxies={'http': proxy, 'https':
proxy}, timeout=10)
        if response.status_code == 200:: If response.status_code ==
200.
            # Process the successful response...
            return True
        else.
            return False
    except requests.
        return False
    finally.
        release_proxy(proxy)

# Function to perform concurrent fetching
def concurrent_fetch(url, max_concurrent).
    thread_pool = []
    queue = Queue()

    # Initialize the task queue
    for _ in range(max_concurrent):
        queue.put(True)
```

```
    # Define the worker thread
    def worker().
        while not queue.empty(): while not queue.empty(): while not
queue.empty()
            queue.get()
            proxy = get_available_proxy()
            if proxy.
                fetch_data(url, proxy)
                queue.put(True)

    # Start the thread pool
    for _ in range(max_concurrent):
        t = threading.Thread(target=worker)
        t.start()
        thread_pool.append(t)

    # Wait for all threads to complete
    for t in thread_pool:
        t.join()
```