import asyncio

import aiohttp

from lxml import html as html\_util

from urllib.parse import urljoin, urlparse

class Crawler:

def \_\_init\_\_(self, initial\_url, depth, no\_more\_then\_y\_in\_parallel=20):

self.initial\_url = initial\_url

self.base\_url = '{}://{}'.format(urlparse(self.initial\_url).scheme,

urlparse(self.initial\_url).netloc)

self.depth = depth

self.seen\_urls = set()

self.session = aiohttp.ClientSession()

self.semaphore = asyncio.BoundedSemaphore(no\_more\_then\_y\_in\_parallel)

async def request\_on\_url(self, url):

print('Request on: {}'.format(url))

async with self.semaphore:

tries = 3

while True:

tries = tries - 1

if tries == 0:

break

try:

async with self.session.get(url, timeout=30, ssl=False) as response:

page\_code = await response.read()

return page\_code

except Exception as e:

print('An exception was caught when trying to get HTML data from the URL {}: {}'.format(url, e))

def find\_urls(self, html):

list\_of\_found\_urls = []

dom = html\_util.fromstring(html)

for href in dom.xpath('//a/@href'):

url = urljoin(self.base\_url, href)

if url not in self.seen\_urls and url.startswith(self.base\_url):

list\_of\_found\_urls.append(url)

return list\_of\_found\_urls

async def single\_extract(self, url):

data = await self.request\_on\_url(url)

list\_of\_found\_urls = set()

if data:

for url in self.find\_urls(data):

list\_of\_found\_urls.add(url)

return url, data, list\_of\_found\_urls

async def multiple\_extract(self, go\_through):

futures = []

results = []

for url in go\_through:

if url in self.seen\_urls:

continue

self.seen\_urls.add(url)

futures.append(self.single\_extract(url))

for future in asyncio.as\_completed(futures):

try:

results.append((await future))

except Exception as e:

print('An exception was caught when trying to wait for the future to finish: {}'.format(e))

return results

def parser(self, data):

dom = html\_util.fromstring(data)

title = dom.xpath('//title')

# print(title[0].text)

if title:

title = title[0].text

return {'title': title}

async def crawl\_start(self):

search\_area = [self.initial\_url]

results = []

for depth in range(self.depth + 1):

load = await self.multiple\_extract(search\_area)

search\_area = []

for url, data, found\_urls in load:

data = self.parser(data)

results.append((depth, url, data))

search\_area.extend(found\_urls)

await self.session.close()

return results

if \_\_name\_\_ == '\_\_main\_\_':

url = 'https://a1.ro'

crawler = Crawler(url, 3)

future = asyncio.Task(crawler.crawl\_start())

loop = asyncio.get\_event\_loop()

loop.run\_until\_complete(future)

loop.close()

result = future.result()

print("##########################")

print('Length of the result is {}'.format(len(result)))

print('A sample of the result is ')

for res in result[: 20]:

print(res)

print("##########################")