به نام خدا

گزارش پروژه فشرده‏سازی تصویر

نام دانشجو

امیر خسروی نژاد

شماره دانشجویی

9831113

سوال اول:

مشکل این است که اگر مثلا فاکتور Cb و Cr یا به طور کلی شدت رنگ پیکسلهای کناری و بلاکهای مجاور بالاتر باشد، تغییر رنگ بسیار زیادی را احساس می‏کنیم و خوشایند نیست. البته این مورد هم که روشنایی خیلی زیاد باشد، تاثیر زیادی دارد چون چشم ما به روشنایی حساس‏تر است.

سوال دوم:

هر چقدر سایز بلاک بیشتر شود، نرخ فشرده‏سازی بالاتر می‏رود ولی خب کیفیت عکس را از دست خواهیم داد. در مقابل، هر چه سایز بلاک هم کاهش یابد،

سوال سوم:

before compression: 120

8 -> 00

34 -> 01

5 -> 1000

6 -> 1001

43 -> 1010

127 -> 1011

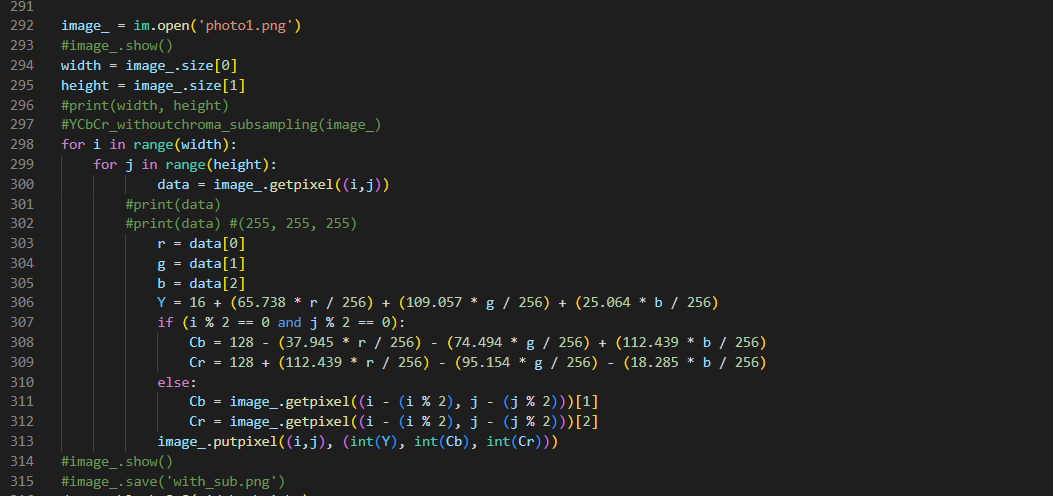
10 -> 11

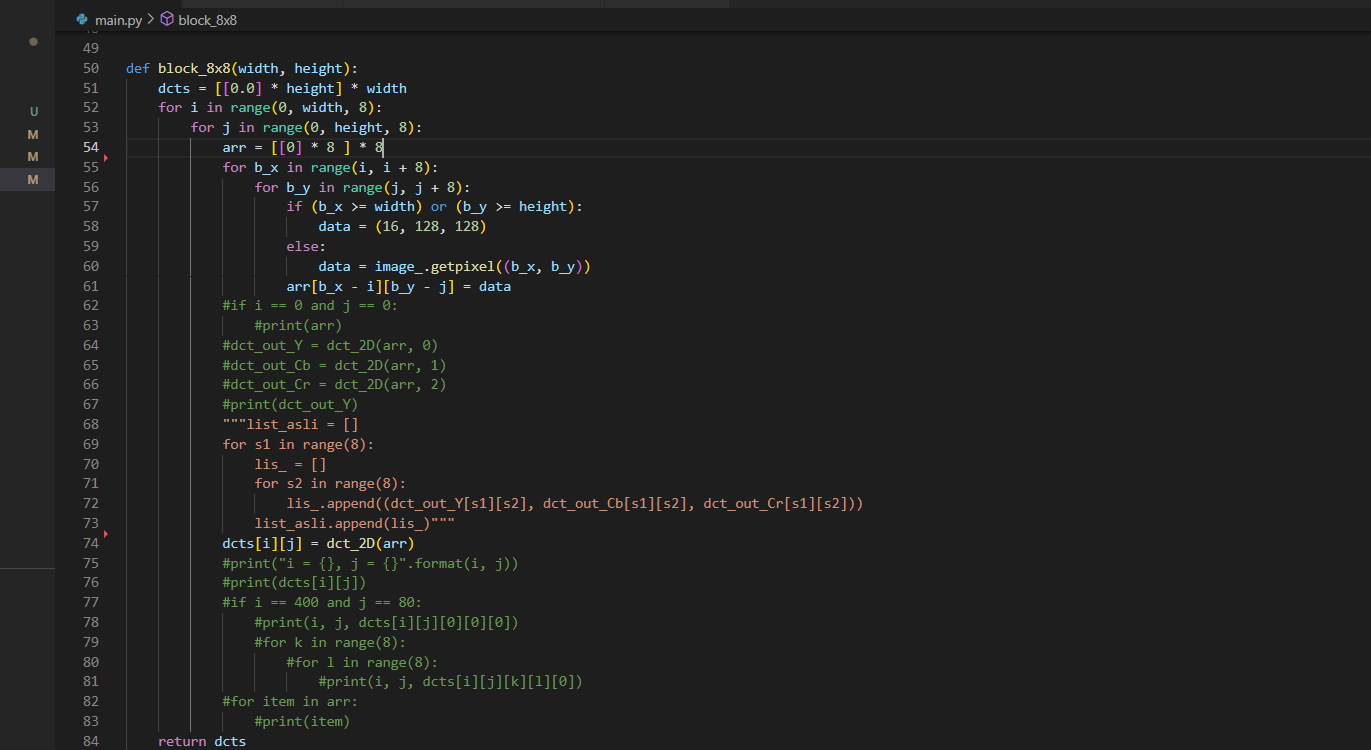
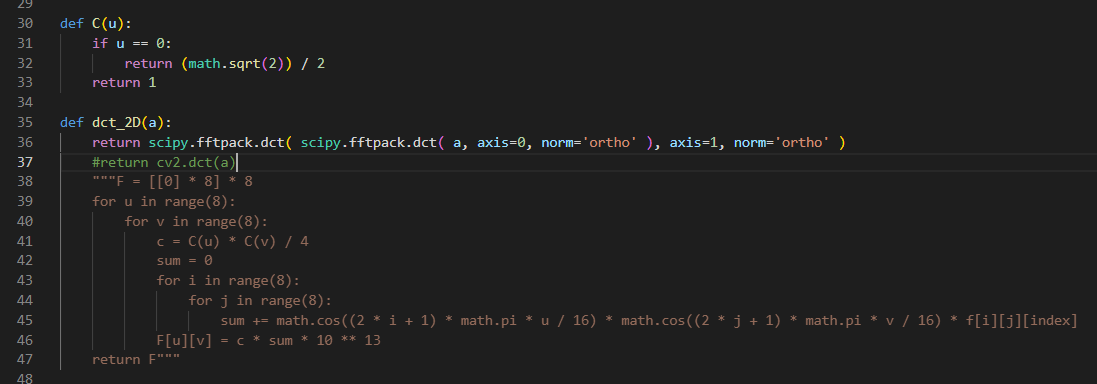
after compression: 38

سوال چهارم:

حجم فایل درخت هافمن مربوط به ضرایب AC حدود 180 بایت و ضرایب DC حدود 2 مگابایت شد!

توضیح مختصر راجع به کد:

همانطور که در تصویر فوق مشخص است، ابتدا عکس اولیه را باز میکنیم. رنگهای هر پیکسل در متغیرهای r, g, b ریخته میشود و طبق فرمول زیر از سیستم rgb به YCbCr میرود. اما نکته‏ای باید در این تکه کد توجه داشت این است که با 4:2:0 subsample صورت گرفته یعنی به طور جزئی در هر 2\*2 در پیکسلها، مقدار Y قبل و بعد از subsampling ثابت و برای Cb, Cr از خانه بالا سمت چپ روی 3 خانه دیگر کپی میشود که پیاده‏سازی عملی آن را در کد می‏بینیم. (در فایلها 2 عکس با فرمت تصویر YCbCr قبل و بعد از subsampling موجود است)

تابع بعدی در بالا مشخص است: این تابع پیمایش 8 واحد 8 واحد روی پیکسلها، بلاکهای 8 تایی را در لیست 2بعدی (arr) میریزد و این مورد که اگر بلاک ناقص ماند، با مشکی پر شود در خط 58 نوشته شده است. بعد از اینکه یک بلوک 8 تایی ساخته شد به تابع dct\_2D داده که با استفاده ار کتابخانه Scipy این تبدیل انجام شده است. در نهایت، بلوکها را در لیست 2بعدی دیگری به نام dcts ذخیره میکنیم و آن را برمیگردانیم. 

در شکل بالا، پیاده‏سازی dct\_2D را مشاهده میکنید. خطوط کامنت شده‏ی قرمز تابع دستی پیاده‏سازی شده با استفاده از فرمول اصلی تبدیل DCT بود که با بررسی نتایج آن، به این جمع‏بندی رسیدم که دقت خیلی بالایی دارد اما این دقت چندان برای ما اهمیت ندارد و علاوه بر آن محاسبات را بسیار کند میکند. ( مثلا در خیلی از بلاکهایی که جواب در روش دستی پیاده سازی شده در مرتبه 10-14 بود، با کتابخانه مقدار صفر میگرفت)

در تابع بعدی که quantization و round انجام میشود، ابتدا جداول luminance و chrominance را آورده و دوباره با پیمایش روی هر بلاک، مقدار فعلی بلاک در سه مولفه Y, Cb, Cr را تقسیم بر خانه مرتبط در جدول مربوطه میکنم و round انجام میدهم. این مورد بلاک 8\*8 جدیدی به ما میدهد که در لیست list\_kl ذخیره میکنم. سپس در تابع زیگزاگ، بردار 64 عضوی از هر بلاک ما را میدهد که 1 عضو اول آن را در بلاکهای جدا (DC) حساب کرده و در یک لیست میریزم. با تابع dpcm به صورت تفاضلی برای هر سه مولفه Y, Cb, Cr ذخیره و حاصل در یک لیست جدا (new\_) ریخته میشود.

نهایتا با تبدیل تک تک عضوها و اعداد به مقدار باینری و کد کردن سایز آنها با hauffman coding آنها را با pickle ذخیره میکنیم.