

الف) همانطور که مشاهده میکنید در خروجی سلول اول، تصویر جدید از لحاظ شدت روشنایی و رنگ ها کاملاً شبیه به تصویر `refrence` شده است. اما ساختار تصویر خروجی کاملاً شبیه به تصویر `source` بوده و تغییر نکرده. در خروجی سلول دوم هم همانطور که انتظار داشتیم، `cdf` تصویر خروجی بر روی `cdf` تصویر `refrence` منطبق شده. دقت کنید که مجموعه همه شدت روشنایی های تصویر خروجی یک زیرمجموعه از مجموعه همه شدت روشنایی های تصویر `refrence` بوده و `cdf` هر شدت روشنایی در تصویر خروجی برابر `cdf` همان شدت روشنایی در تصویر `refrence` است.

ب) مسایلی که دوست داریم از لحاظ رنگی یک تصویر را مشابه یک تصویر دیگر کنیم. مسایلی که به دلیل عکس برداری باعث شده که تصویر ما رنگش را آن طور که باید حفظ نکرده و دوست داریم ترکیب رنگی آن را شبیه به حالت اصلی خودش کنیم.

توضیح کد)

توضیح تابع `calc_hist`:

کل عکس را سرچ کرده و هنگامی که پیکسلی با شدت روشنایی `i` دیدیم، مقدار آن را به هیستوگرام مربوطه اضافه کرده. پیچیدگی این الگوریتم  $O(w * h)$  می‌باشد که `w` و `h` ابعاد تصویر ورودی است.

توضیح تابع `calc_cdf`:

چون میتوان این قسمت را به صورت بازگشتی حل کرد، آن را با `dynamic programming` حل کرده ایم. ابتدا هیستوگرام را حساب کرده و سپس با `dp` تابع توزیع تجمعی عکس را پیدا کرده ایم. پیچیدگی این الگوریتم  $O(w * h)$  می‌باشد که `w` و `h` ابعاد تصویر ورودی است.

توضیح تابع `hist_matching`:

اگر بخواهم مقداری درک کد زده شده را ساده تر کنم, باید بگویم که این قسمت مانند قسمت ادغام الگوریتم merge sort هست. به این صورت که دو پوینتر داریم  $i$  و  $j$ , که میبینیم  $cdf$  پیکسل  $i$  در تصویر اولیه با  $cdf$  پیکسل  $j$  در تصویر مرجع برابر شده و چون  $cdf$  یک تابع صعودی است, به سمت جلو حرکت کرده تا برای همه  $i$  ها که در تصویر اولیه موجود هستند, مقداری که نزدیک است  $cdf$  آن در تصویر مرجع را بیابیم.