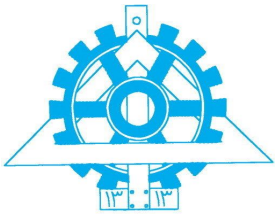
# تمرین کامپیوتری سوم



دانشکده مهندسی برق و کامپیوتر

**گزارش کار**

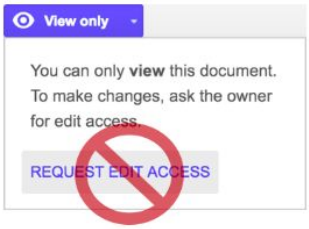
سیستم‌های عامل - پاییز ۱۳۹۹

استاد:
**دکتر مهدی کارگهی**

**تاریخ:**
**۲۴ آذر ماه ۱۳۹۹**

نام و نام خانوادگی:
مهیار کریمی

---

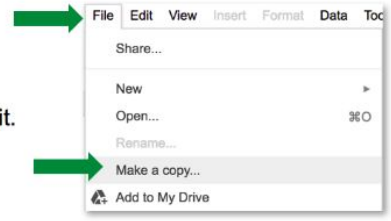



**How to use this template:**

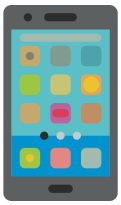
This is a view-only file and cannot be edited.

**Create your own copy** of this template to edit.

In the menu, click **File > Make a copy...**



## مقدمه



در این تمرین به تحلیل داده‌هایی که از مشخصات و قیمت فروش گوشی‌های موبایل جمع‌آوری شده‌است پرداخته شده است. در ابتدا برنامه اقدام به خواندن و تجزیه مجموعه داده[1]ی ارائه شده می‌کند و آنها را در حافظه خود ذخیره می‌کند. پس از استخراج داده‌ها و ویژگی‌های آنها، برنامه اقدام به نرمال‌سازی[2] داده‌ها و در نهایت اقدام به تعیین طبقه قیمتی گوشی‌ها می‌کند. این تمرین به دو روش این مسئله پیاده‌سازی شده است که در ادامه گزارش، نتایج حاصل آمده است.


[1] Dataset

[2] Data Normalization

# پیاده‌سازی سری

## سوال اوّل

**چرا برای پیاده‌سازی یک برنامه بصورت چندریسه‌ای، بهتر است ابتدا این برنامه بصورت سری پیاده‌سازی شود؟**

جواب: پیاده‌سازی و اشکال‌زدایی (Debugging) برنامه بصورت سری آسانتر از پیاده‌سازی چندریسه‌ای است و برای بررسی درستی الگوریتم مورد نظر، روش بهتری نسبت به پیاده‌سازی چندریسه‌ای می‌باشد. هم‌چنین پس از پیاده‌سازی سری، تشخیص بخشهایی از کد که می‌توانند بصورت موازی اجرا شوند سریعتر انجام می‌شود.

## سوال دوم

**با بررسی زمان اجرای بخش‌های مختلف برنامه، Hotspot[3] های برنامه را مشخص کنید.**

جواب: می‌دانیم برای یک پردازه تک‌ریسه‌ای، در خروجی برنامه time رابطه زیر برقرار است:

$$T_{real} = T_{user} + T_{sys}$$

با بررسی خروجی time در چند اجرای متوالی، سهم $T_{user}$ بیشتر از $T_{sys}$ است؛ پس می‌توانیم نتیجه بگیریم که برنامه ما CPU-Bound می‌باشد و بیشتر زمان اجرای برنامه صرف انجام محاسبات و تجزیه فایل‌های ارائه شده می‌گردد.

در پیاده‌سازی ما، توابع زیر بیشترین زمان اجرا را به خود اختصاص می‌دهند:

| وظیفه تابع | نام تابع |
|---|---|
| خواندن و تجزیه نمونه‌های ورودی در فایل train.csv | Evaluator::read_data |
| نرمال‌سازی مقدار featureها | Evaluator::normalize_data |
| دسته‌بندی هر نمونه ورودی و بررسی عملکرد مدل به ازای ورودی مورد نظر | Evaluator::evaluate_model |

[3] توابعی که در برنامه‌تان بیشترین زمان اجراها را به خود اختصاص می‌دهند.

### جدول اوّل

**زمان‌های اجرای ۶ اجرای متوالی از برنامه و میانگین آن‌ها را بازای ورودی نمونه‌ای که در شرح تمرین آمده است، در جدول زیر بیاورید.**

| اجرای اوّل | اجرای دوم | اجرای سوم | اجرای چهارم | اجرای پنجم | اجرای ششم | میانگین |
|---|---|---|---|---|---|---|
| 0.039 | 0.039 | 0.042 | 0.040 | 0.039 | 0.040 | 0.0398 |

توضیح: مقادیر بالا در مقیاس ثانیه هستند.

## پیاده‌سازی چندریسه‌ای

### سوال سوم

**اگر هنگام موازی‌سازی برنامه به زمان اجرای بیشتری نسبت به حالت سری برخورد کنید، چه رویکردهایی را برای کاهش زمان اجرا و استفاده حداکثری از موازی‌سازی پیش می‌گیرید؟**

جواب: برای اینکه ریسه‌ها بتوانند به صورت همزمان از منابع سیستم استفاده کنند، تعداد آن‌ها را تا حد امکان کم نگه می‌داریم (افزایش تعداد ریسه‌ها باعث بوجود آمدن سربار اجرا برای تعویض متن بین ریسه‌ها و زمان‌بندی آنها می‌شود، زیرا تعداد ریسه‌هایی که بطور همزمان اجرا می‌شوند حداکثر برابر تعداد هسته‌های پردازشی منطقی سیستم است.) همچنین، تنها بخشهایی به ریسه‌ها اختصاص یافته‌اند که قابل موازی‌سازی باشند و محاسبات روی مجموعه‌های کوچک داده به صورت سری انجام شده‌است.

### سوال چهارم

**در هنگام پیاده‌سازی این بخش، به چه چالش‌هایی برخورد کردید و بیان کنید که به چه صورت آن‌ها را رفع کردید.**

جواب: می‌دانیم که همه دسته‌بندها از مجموعه یکتایی از بردارهای وزن برای دسته‌بندی استفاده می‌کنند؛ به همین دلیل، پس از یک بار خواندن فایل بردارهای وزن، برای ارزیابی مدل به ازای هر ریسه، اطلاعات مربوط به این بردار نیز با ارجاع پاس داده می‌شوند. همچنین، برای پیشگیری از استفاده متغیرهای global، استراکچری با عنوان Parallel_Arg ساخته‌ایم که همه اعضای آن بصورت رفرنس هستند. تعریف این استراکچر به صورت زیر است:

```
struct Parallel_Arg {
    const std::vector<std::vector<double> >& weights;
    std::pair<int, int>& total;
    std::string dataset_dir;
    int idx;
    pthread_mutex_t& mutex;
    Evaluator pe;
};
```

## سوال پنجم

**با توجه به تجربه‌ای که در پیاده‌سازی این تمرین بدست آوردید، به نظر شما در چه مواقعی از قفل[4] در یک طراحی چندریسه‌ای ضروری است؟ تاثیر استفاده از قفل‌ها را بر روی کارآیی[5] سامانه بیان کنید.**

جواب: استفاده از قفل تنها زمانی ضروری‌است که بخواهیم مقدار متغیری را توسط چند ریسه تغییر دهیم؛ برای مثال در پیاده سازی ما، زوج مرتب total برای نگه داشتن تعداد کل نمونه‌ها و تعداد حدس‌های درست به کار رفته‌است. هر ریسه پس از پایان محاسبه روی فایل متناظر خود، این مقدار را بروز رسانی می‌کند. برای جلوگیری از ایجاد اشتباه در محاسبات، از یک قفل برای مشخص کردن بخش مربوط به بروز رسانی این متغیر استفاده کرده‌ایم. با توجه به این که سایر بخش‌های داده یا فقط توسط هر ریسه خوانده می‌شوند و یا بطور محلی برای همان ریسه هستند، استفاده از قفل در سایر مراحل محاسبه ضروری نیست.

با قفل کردن قسمتی از کد، همه ریسه‌ها (بجز ریسه‌ای که در حال حاضر در حال اجرا می‌باشد یا قفل را فعال کرده‌است) برای اجرای کد قفل شده باید منتظر بمانند تا ریسه قفل کننده اجرای محدوده بحرانی (Critical) را به پایان برساند. به همین دلیل، استفاده از قفل سبب اضافه شدن به سربار اجرای برنامه ما شود و بایستی از قفل‌ها بطور محدود و بهینه استفاده کنیم.

## جدول دوم

**زمان‌های اجرای ۶ اجرای متوالی از برنامه و میانگین آن‌ها را بازای ورودی نمونه‌ای که در شرح تمرین آمده است، در جدول زیر بیاورید.**

[4] Lock

[5] Performance

| اجرای اوّل | اجرای دوم | اجرای سوم | اجرای چهارم | اجرای پنجم | اجرای ششم | میانگین |
|---|---|---|---|---|---|---|
| 0.29 | 0.25 | 0.31 | 0.27 | 0.26 | 0.27 | 0.0275 |

**میزان تسریع ( $\frac{Serial\ Time}{Parallel\ Time}$ ) برنامه نسبت به حالت سری را در زیر بیاورید.**

| میانگین زمان اجرای سری | میانگین زمان اجرای موازی | میزان تسریع |
|---|---|---|
| 0.0398 | 0.0275 | 1.45 |