

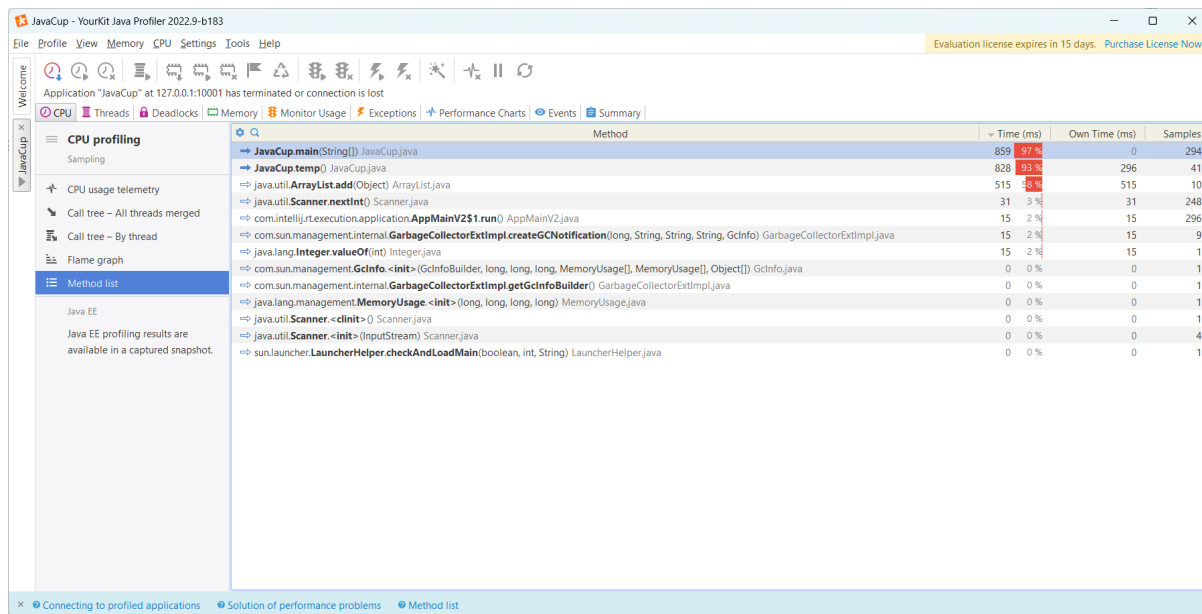
بسمه تعالی

گزارش آزمایش شماره 4 مهندسی نرم افزار

محمدحسین قیصریه 97106238

محمد حیدری 97110071

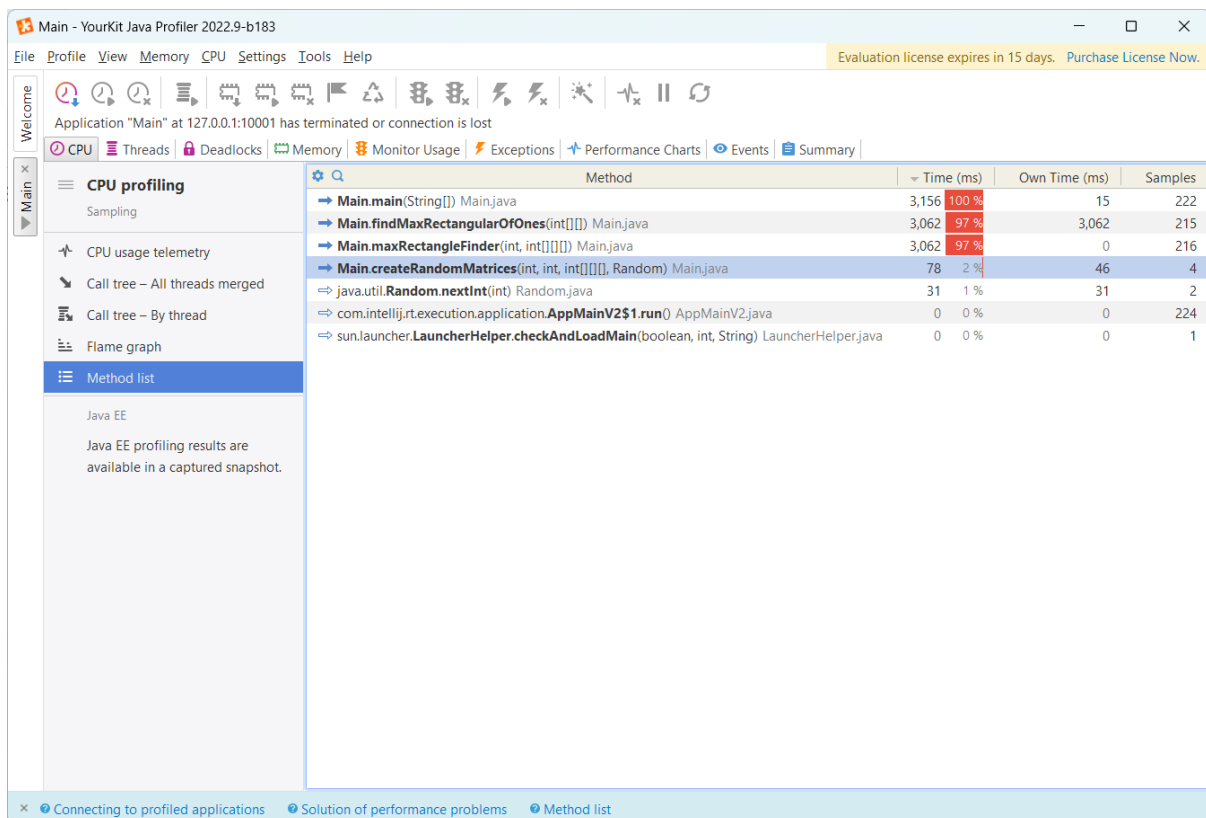
(بخش اول)



با توجه به گزارش YourKit بیشترین مصرف را تابع add کردن آرایه‌ها دارد که ۵۱۵ میلی‌ثانیه زمان مصرف می‌کند.

بخش دوم) برنامه‌ای را در نظر بگیرید که:

- در ابتدا تعداد زیادی ماتریس باینری مربعی تولید کند.
 - سپس در هر کدام اندازه بزرگترین مستطیل شده از یک را پیدا و اعلام کند.
- ابتدا با الگوریتم brute force این مسئله را حل می‌کنیم. الگوریتم brute force ما در $O(n^6)$ این مقدار را پیدا می‌کند. برنامه مربوطه در فولدر OneMatrixBruteForce پیاده‌سازی شده است. برنامه ما ابتدا ۱۰۰,۰۰۰ ماتریس 10×10 رندوم در یک تابع ایجاد می‌کند و سپس این ماتریس‌ها را به تابع پیدا کننده بیشترین سائز ارجاع می‌دهد. توابع چاپ کننده حاصل را نیز کامنت کردیم که فقط توابع محاسباتی در برنامه دخیل باشند.



The screenshot shows the YourKit Java Profiler interface. The 'CPU profiling' section is active, displaying a table of methods and their execution statistics. The table has columns for Method, Time (ms), Own Time (ms), and Samples. The methods listed are:

Method	Time (ms)	Own Time (ms)	Samples
Main.main(String[]) Main.java	3,156	100 %	15
Main.findMaxRectangularOfOnes(int[][]) Main.java	3,062	97 %	3,062
Main.maxRectangleFinder(int, int, int[][]) Main.java	3,062	97 %	0
Main.createRandomMatrices(int, int, int[][], Random) Main.java	78	2 %	46
java.util.Random.nextInt(int) Random.java	31	1 %	31
com.intellij.rt.execution.application.AppMainV2\$1.run() AppMainV2.java	0	0 %	0
sun.launcher.LauncherHelper.checkAndLoadMain(boolean, int, String) LauncherHelper.java	0	0 %	0

۳۰۶۲ میلی‌ثانیه برای یافتن بزرگترین ماتریس واحد زمان مصرف شده است. و ۷۸ میلی‌ثانیه برای ایجاد ماتریس‌های رندوم. فلذا باید تابع یافتن بزرگترین ماتریس واحد را بهبود بخشید.

فلذا تابع مربوط به یافتن بزرگترین ماتریس واحد را بهینه می‌کنیم و با استفاده از برنامه‌نویسی پویا این تابع را در $O(n^2)$ بازنویسی می‌کنیم. پس از پروفایل کردن داریم:

The screenshot shows the YourKit Java Profiler interface. The left sidebar displays the 'CPU profiling' section with various views like 'Sampling', 'CPU usage telemetry', 'Call tree', 'Flame graph', 'Hot spots', and 'Method list'. The main window shows a table of methods being profiled:

Method	Time (ms)	Own Time (ms)	Samples
→ Main.main(String[]) Main.java	265 100 %	31	24
→ Main.findMaxRectangularOfOnes(int[][]) Main.java	125 47 %	125	12
→ Main.maxRectangleFinder(int, int[][]) Main.java	125 47 %	0	12
→ Main.createRandomMatrices(int, int, int[][], Random) Main.java	109 41 %	62	9
⇒ java.util.Random.nextInt(int) Random.java	46 17 %	46	4
⇒ com.intellij.rt.execution.application.AppMainV2\$1.run() AppMainV2.java	0 0 %	0	34
⇒ com.sun.management.internal.GarbageCollectorExtImpl.createGCNotification(long, String, St	0 0 %	0	2
⇒ sun.launcher.LauncherHelper.checkAndLoadMain(boolean, int, String) LauncherHelper.java	0 0 %	0	1

Below the table, there are tabs for 'Back Traces', 'Callee List', and 'Merged Callees'. The 'Back traces' tab is active, showing a 'Reverse Call Tree' for the selected method 'Main.main'. It shows a single entry for 'Main.main' with 265ms and 100% of the time.

A dialog box titled 'Send Anonymous Usage Statistics' is visible in the bottom right corner, asking if the user wants to help YourKit developers by sending anonymous usage statistics. The dialog includes a 'Yes, I Want to Help' button and a 'No' button.

همان‌طور که شاهد هستید ۶۲ ثانیه برای ایجاد ماتریس‌های رندوم مصرف شده و ۱۲۵ ثانیه برای یافتن بزرگترین ماتریس واحد زمان مصرف شده. فلذا برنامه با بهینه کردن تابع یافتن بزرگترین ماتریس واحد، به کلی بهینه شد.