بسم الله الرحمن الرحیم

آزمهندسی نرم افزار

آزمایش اول

سید مهدی فقیه 97106198

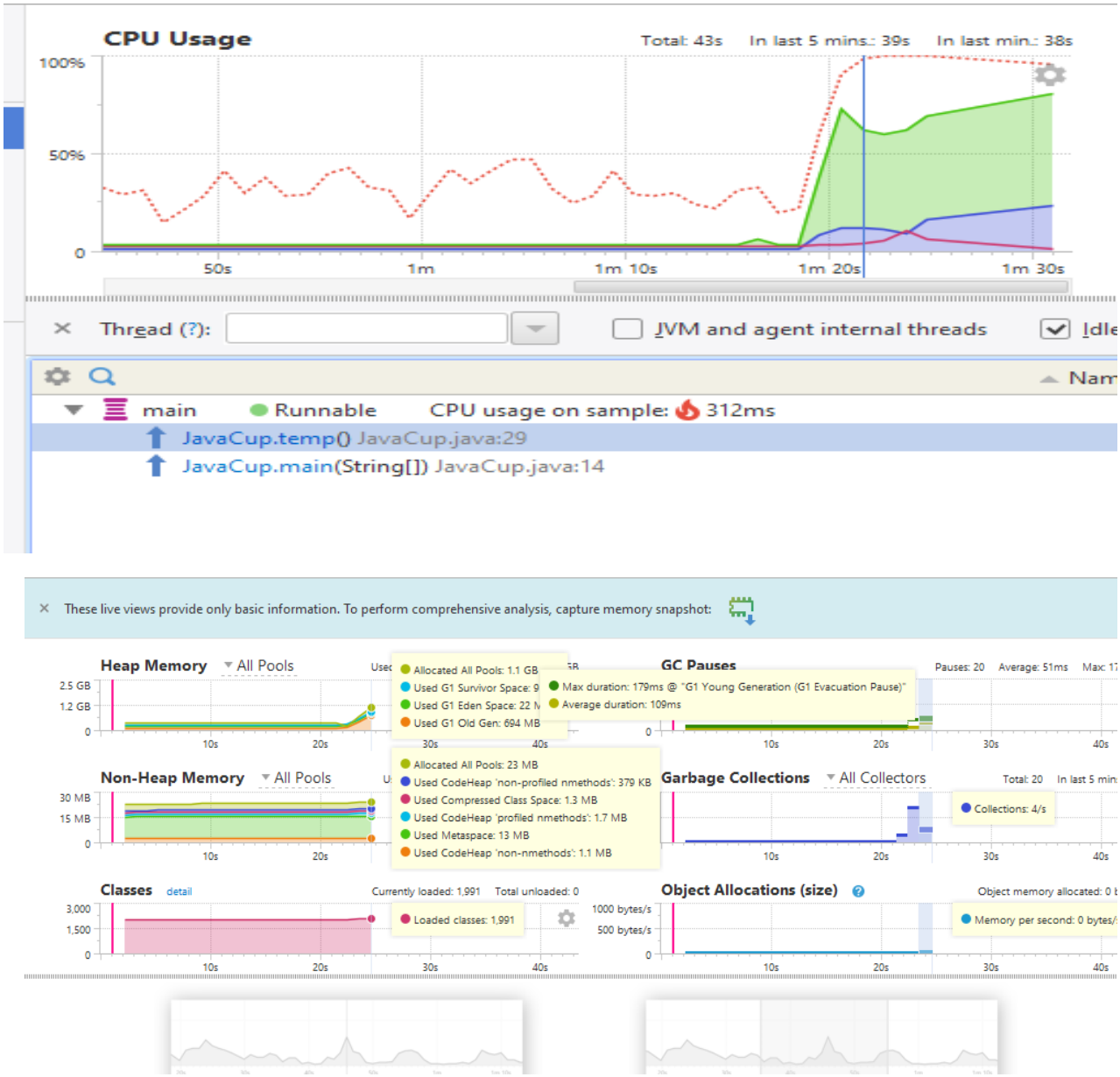
محمدرضا یوسف پور 97106324

لینک گیت : https://github.com/SMahdiFaghih/SoftwareEngineeringLab-E1

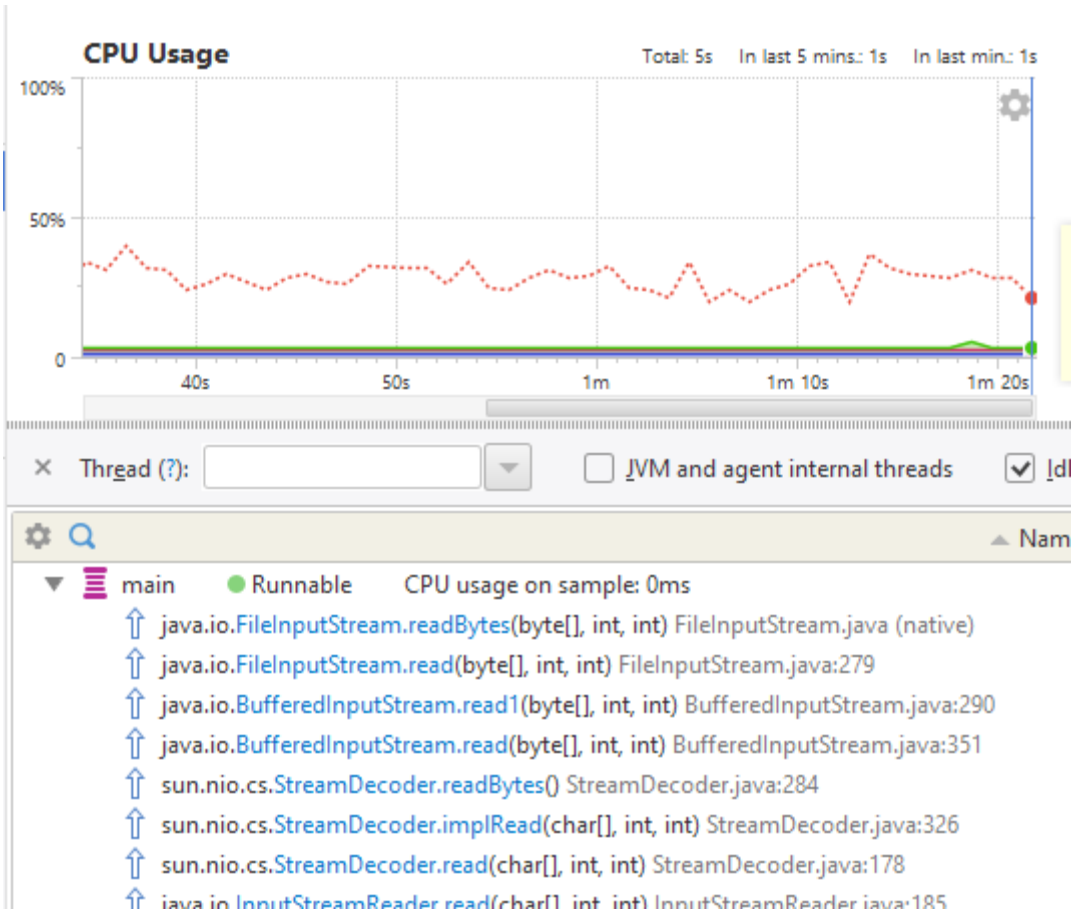
شایان ذکر است که هر دو بخش آزمایش در لینک بالا موجود است.

هر دو اعضای گروه مشارکت یکسانی در انجام آزمایش داشتیم.

پس از profile کردن کلاس java cup با استفاده از برنامه ی yourkit ، ابتدا تابع main در حال اجرا شدن بود. در ابتدای کار تابع main بیشتری مصرف cpu را داشته است چون برنامه منتظر ورودی گرفتن از کاربر است. java.io.FileInputStream.readBytes(byte[], int, int) بیشترین مصرف را دارد بعد از آن که از کاربر ورودی دریافت کرد java.io.FileInputStream.read(byte[], int, int) بیشترین مصرف cpu را دارد. همانطور که در کد مشاهده می کنید پس از گرفتن سه ورودی از کاربر توابع temp و eval را اجرا میشوند که تابع temp بیشترین مصرف cpu را دارد چرا که با توجه به تابع temp یک arraylist ساخته می شود و در داخل دو حلقه ی تو در تو یکی به اندازه ی 10000 و دیگری به اندازه ی 20000 عملیات جمع کردن و افزودن به لیست را انجام می دهد که ان تابع مصرف بسیار زیادی از cpu را دارد برای سیستم من حدود 95 درصد از سیستم را درگیر می کند. تابع eval قرار است بعد از تابع temp اجرا شود اما با توجه به سنگین بودن تابع temp اگر اروری مبنی بر memory limit دریافت نکنیم پس از زمان بسیار زیادی تابع eval اجرا می شود و اگر هم که ارور دریافت کنیم برنامه terminate می شود و تابع eval اجرا نمی شود.

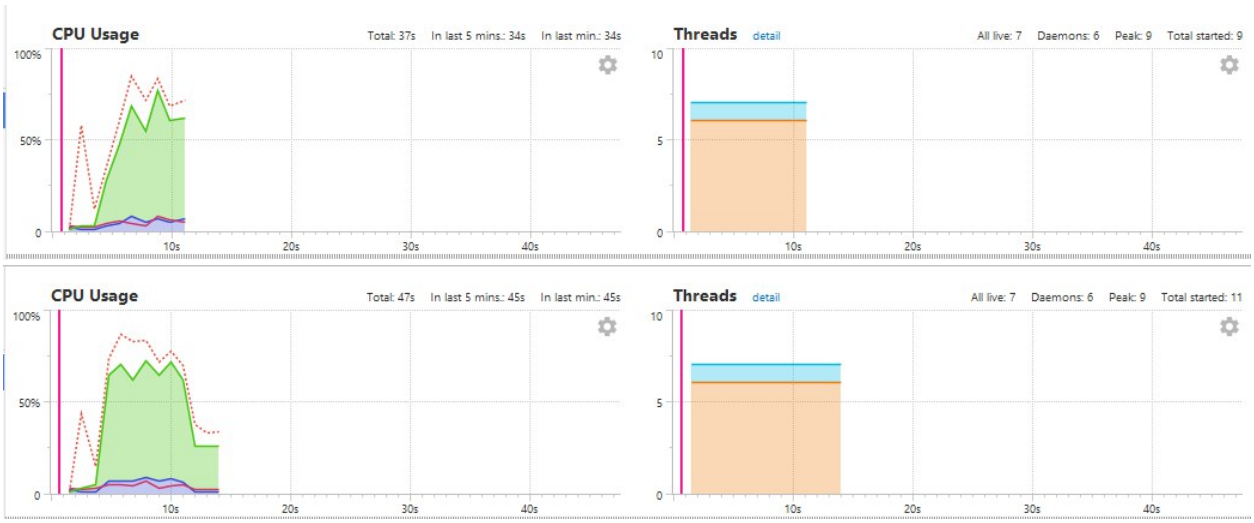


با توجه به عملکرد تابع temp می توانیم به صورت کلی این تابع را حذف کنیم چرا که اجرای این تابع هیچ منفعتی برای ما ندارد و همچنین هیچ استفاده ای از خروجی این تابع نمیشود. اگر این تابع را حذف کنیم و مجددا عملیات profiling را اجرا کنیم پس از دریافت ورودی ها تابع مقدار مصرفی cpu بسیار کم می شود و می توان گفت که تابع eval مصرف بسیار کمی از cpu را دارد که کاملا حتی در مقایسه ی با مصرف input / output به شدت قابل چشم پوشی است.



اگر هم قرار نباشد که این تابع را حذف کنیم آن را باید به صورت زیر تغییر دهیم تا مصرفcpu و منابع بهتر شود.

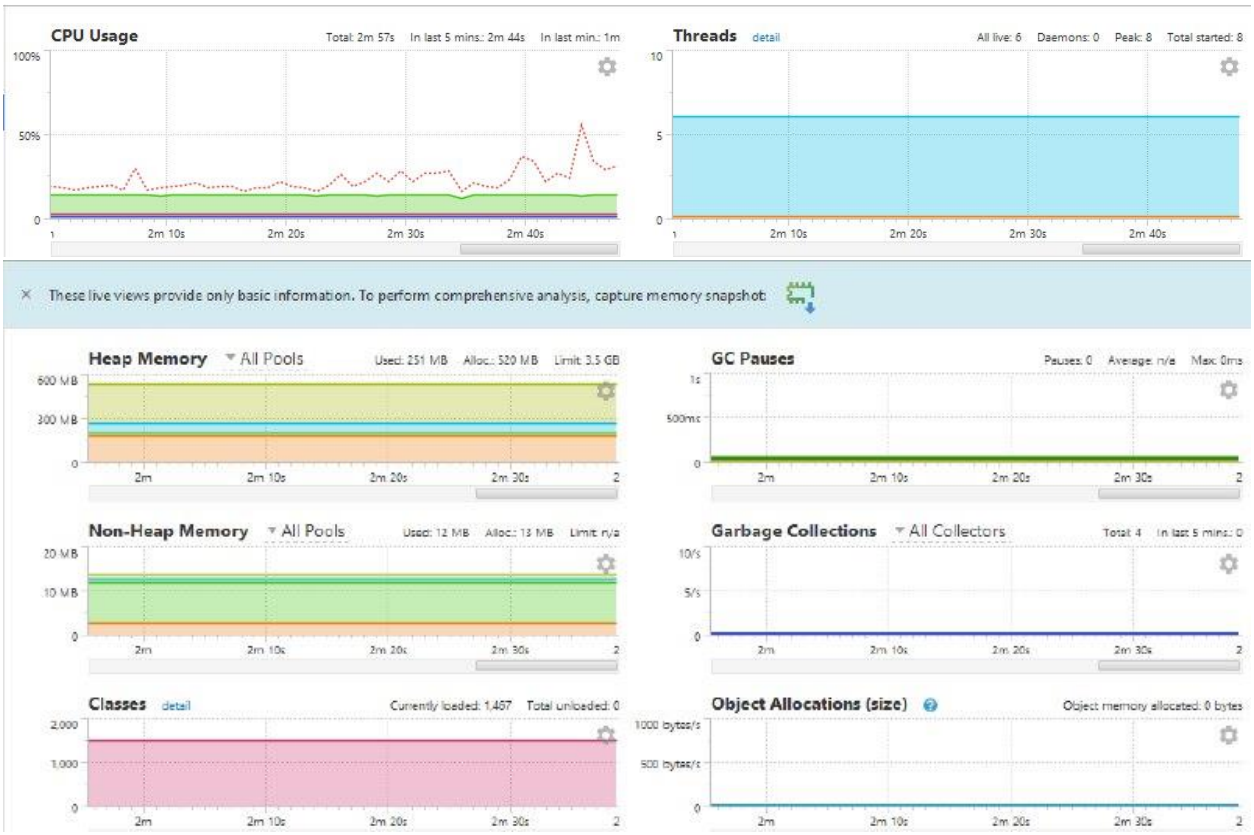
یک راهی که برای تغییر مقدار مصرفی cpu به ذهنمان رسید این است که در ابتدای کار مقدار capacity ، arraylist را مشخص کنیم و این مقدار را برابر با 10000*20000 قرار دهیم که با این تغییر مصرف cpu از 76 درصد به 71 درصد کاهش یافت که این نشان دهنده ی این است که اگر این کار را انجام ندهیم پس از گذشت مدتی به ارور out of memory بر میخوریم چون که در هر بار insert اگر arraylist پر شده باشد یک arraylist جدید با سایز 1.5 برابر ساخته می شود اما اگر این مقدار را برای arraylist در ابتدا ست کنیم دیگر نیازی به 1.5 برابر شدن سایز نیست که نتیجه ی این تغییر را در تصاویر زیر مشاهده می کنید.



تصویر اول برای قبل از اعمال تغییرات است ، تصویر دوم پس از اعمال است.

– 2

در تصویر اول همانطور که می‌بیند زمان اجرای برنامه برای ده میلیون عدد در بدترین حالت (در حالت نزولی) حدود 2 دقیقه و 40 ثانیه طول کشیده است که نشاندهنده ی بد بودن الگوریتم سورت است الگوریتم استفاده شده در دو تصویر اول الگوریتم selection sort است که این الگوریتم مدنظر ما نیست و باید آن را بهبود دهیم و از الگوریتم merge sort استفاده کنیم که در ادامه تصاویر مربوط به الگوریتم بهتر را مشاهده می کنید.



در دو تصویر زیر همانطور که مشاهده می کنید زمان اجرای merge sort برای 10 میلیون عدد در حالت نزولی به شدت کم تر و سرعت اجرا بیشتر شده است که این نشان می دهد که ما توانستیم الگوریتم را به نحو خوبی تغییر دهیم تا سرعت اجرای برنامه بهتر شود.