به نام خدا

گزارش کار پروژه ی دوم درس ساختار داده ها

تشخیص ساختارهای اجتماعی در گراف

امیرمحمد پیرحسین لو

۹۵۳۱۰۱۴

چکیده:

در این پروژه هدف ا صلی تشخیص ساختارهای اجتماعی در گراف ها ا ست که با ا ستفاده از روش های مختلفی این عمل امکان پذیر است . در ادامه ، روشی که در صفحه ۱۳ مقاله ذکر شده ، پیاده سازی شده است . این روش را می توان با الگوریتم ها متفاوتی پیاده سازی کرد که این الگوریتم ها از نظر زمان و میزان حافظه ی م صرفی با یک دیگر مقایسه شده اند و نمودارهای به د ست آمده از این مقایسه ها تر سیم شده اند که می توانند ما را در یافتن بهترین الگوریتم ها یاری کنند.

مقدمه:

تعریف ساختارهای اجتماعی در یک گراف :

در گراف های بزرگ ، اگر نواحی ای وجود دا شته با شند که تراکم یال ها در آن نواحی نسبت به نواحی دیگر بیشتر باشد (به عبارتی یال های آن ناحیه ارتباط بیشتری با هم داشته باشند) ، هر کدام از این نواحی یک ساختار اجتماعی محسوب می شوند .

کاربرد تشخیص اجتماع ها:

تشخیص ساختارهای اجتماعاتی یک موضوع مهم در بسیاری از زمینه ها می باشد. این موضوع با مفاهیمی مانند شبکه های اجتماعی(روابط بین اعضا،) تحقیقات بیولوژیکی یا مسایل تکنولوژیکی(بیهنه سازی زیرساختهای حجیم) در ارتباط می باشد .

الگوریتم ها و ساختار داده های استفاده شده:

۱- نحوه پیاده سازی:

پیاده سازی این برنامه با استفاده از زبان برنامه نویسی Java و کتابخانه های log4j و apache.tomcat صورت گرفته اســـت. از کتابخانه log4j برای ثبت اطلاعات خروجی در یک فایل و از کتابخانه apache.tomcat برای نمایش نمودار میزان حافظه مصرفی الگوریتم در هر لحظه در برای هر test case استفاده شده است.

۲- ساختمان داده های استفاده شده:

۲,۱- برای ذخیره گراف در ماتریس مجاورت از ساختارهای زیر استفاده شده است:

یک آرایه ۲ بعدی برای ذخیره یال ها به صورت sparse matrix ،

یک آرایه یک بعدی برای ذخیره درجه هر راس که در ذخیره سـازی به فرم لیسـت مجاورت هم مورد اسـتفاده قرار می گیرد.

یک ارایه یک بعدی به اسم startPoints که شروع هر راس در sparse ماتریس در آن قرار دارد و با O(1) محل آن را به ما می دهد.

یک آرایه یک بعدی به اســم edges که شــامل یال های گراف در قالب کلاس Edge اســت که هم برای حالت ماتریس مجاورت و هم لیست مجاورت مورد استفاده قرار می گیرد.

۲,۲- برای ذخیره گراف در لیست مجاورت از ساختارهای زیر استفاده شده است:

یک ارایه یک بعدی که هر خانه آن آدرس شروع یک Linked List است . در هر خانه ی آرایه یک راس وجود دارد و لیست پیوندی متناظر با آن نشان دهنده راس های مجاور آن راس است .

۲,۳- برای استفاده از الگوریتم BFS از ساختار داده Queue استفاده شده است.

۳- الگوریتم ها استفاده شده:

۳,۱- الگوریتم تشخیص دور به طول ۳:

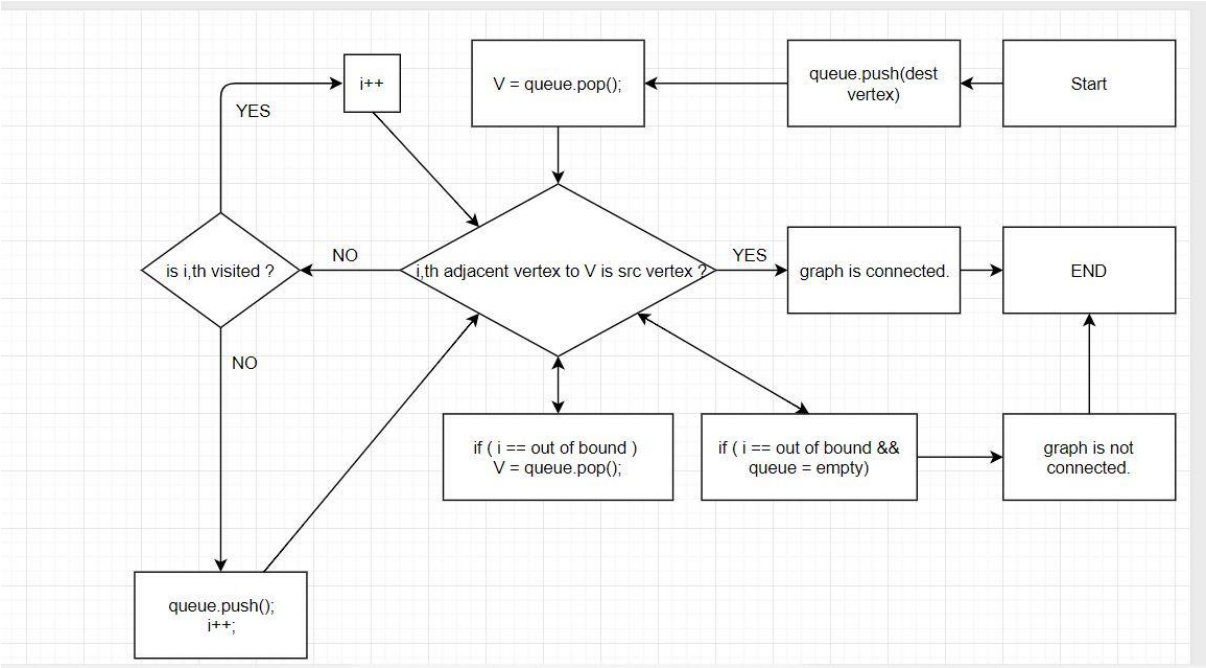
الگوریتمی که با استفاده از دو حلقه تعداد دور های به طول ۳ گذرا از یک راس خاص را به ما می دهد .

یک مثلث شامل سه راس a , b , c را در نظر بگیرید که میخواهیم دور های به طول سه که گذرا از a , b هستند را بیابیم . در این الگوریتم تمام اطرافیان b را مورد بررسی قرار داره و اگر هر کدام مجاور راس a بودند ، یک دور به

تعداد دور ها ا ضافه می شود . به فرض درجه میانگین راس ها k با شد . مرتبه زمانی این الگوریتم در حالت میانگین برابر $O(k^2)$ می شود و پیچیدگی از لحاظ حافظه مصرفی برابر $O(1)$ است .

۳,۲- الگوریتم تشخیص همبندی گراف:

در هر مرحله که یال حذف می شود ، با شروع از یکی از راس های یال حذف شده (راس مقصد) سعی می کنیم با الگوریتم BFS به راس دیگر (راس مبدا) برسیم . در صورت رسیدن به راس مبدا نتیجه می گیریم گراف همبند است (فرض بر این است که گراف اولیه همبند است.) در غیر این صورت نتیجه می گیریم که گراف ناهمبند است . مرتبه زمانی این الگوریتم در بدترین حالت $O (E + V)$ است که باید کل گراف را پیمایش کنیم. از نظر حافظه مصرفی نیز در queue نهایتا تمام راس ها قرار می گیرند و از طرفی آرایه visited دارای طول V است پس در کل پیچیدگی از نظر حافظه $O(2V) = O (V)$ است .



۳,۳- الگوریتم های مرتب سازی:

برای مرتب سازی از الگوریتم های Insertion Sort , Bubble Sort , Merge Sort , Quick Sort و Optimum Quick Sort ا ستفاده شده ا ست . مقدار N بهینه برای Optimum Sort با آزمایش به د ست می آید .

	Mine	Friend
CPU Model	Intel(R) core (TM) i7-7500U	
CPU Physical Core	2	
CPU Virtual Core	4	
CPU L1 Cash (Bytes)	512	
CPU L2 Cash (Bytes)	512	
CPU L3 Cash (Bytes)	4096	
RAM Model		
RAM Capacity (GB)	16	
RAM Bus		
HDD/SSD Write Speed	30 MB/s	
HDD/SSD Read Speed	30 MB/s	
OS	Windows 10 Enterprise	

Name	Vertex Number	Edge Number	Average Vertex Degree
t1			
t2			
t3			
t4			
t5			
t6			
t7			
t8			

۱- تفاوت استفاده از ماتریس همسایگی و لیست همسایگی در زمان و حافظه :

1-

t2-LinkedList-Optimum-Insertion-24

Time for creating graph in LinkedList format: 266 mili seconds.

11167 Edges are being deleted with this case.

Total time of executing algorithm: 4995 seconds.

Total time of program life cycle: 4995 seconds.

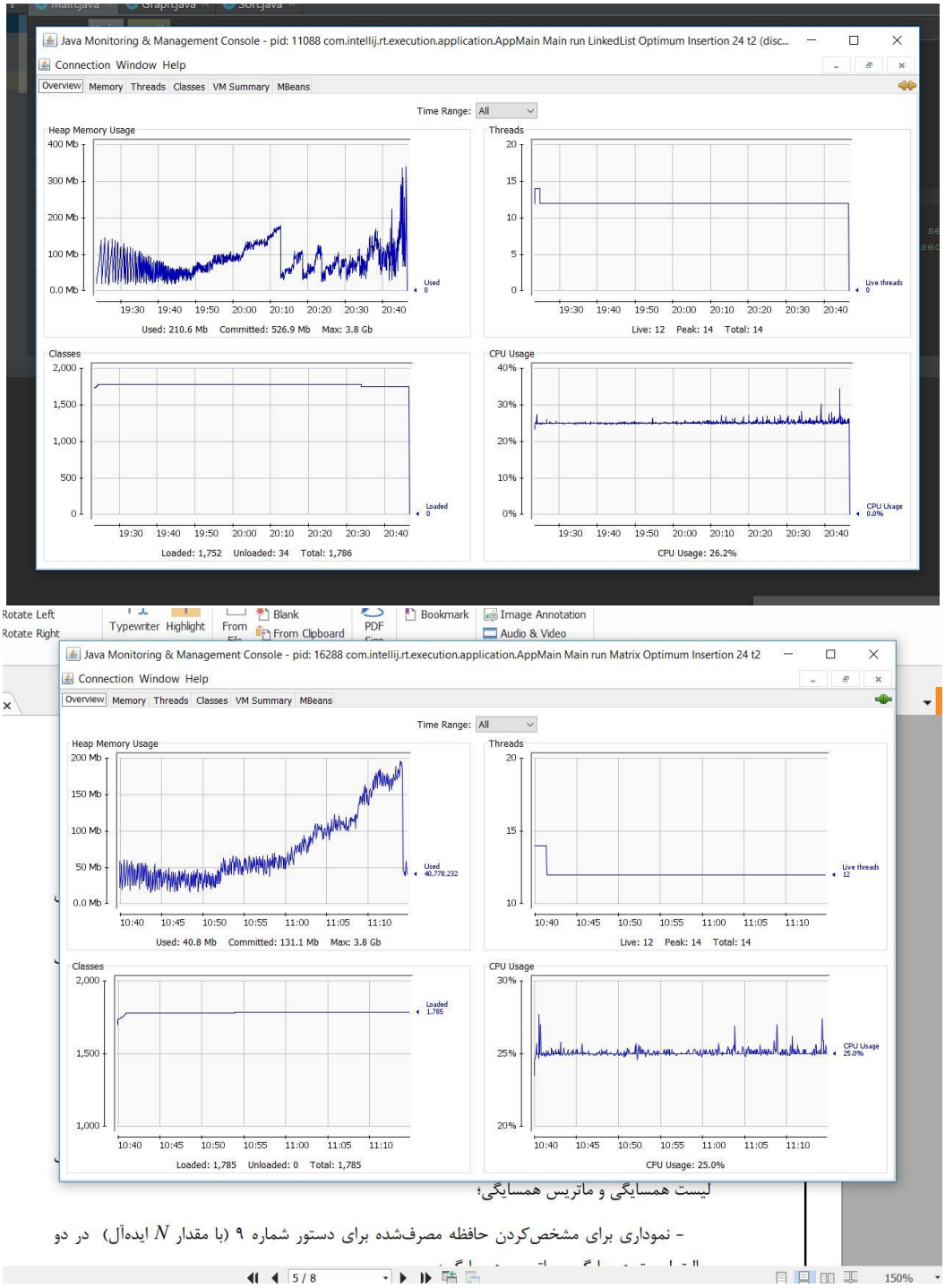
t2-Matrix-Optimum-Insertion-24

Time for creating graph in Matrix format: 625 mili seconds.

10963 Edges are being deleted with this case.

Total time of executing algorithm: 4580 seconds.

Total time of program life cycle: 4581 seconds.



t1-linkedList-Merge

Time for creating graph in LinkedList format: 216 mili seconds.

5767 Edges are being deleted with this case.

Total time of executing algorithm: 644 seconds.

Total time of program life cycle: 644 seconds.

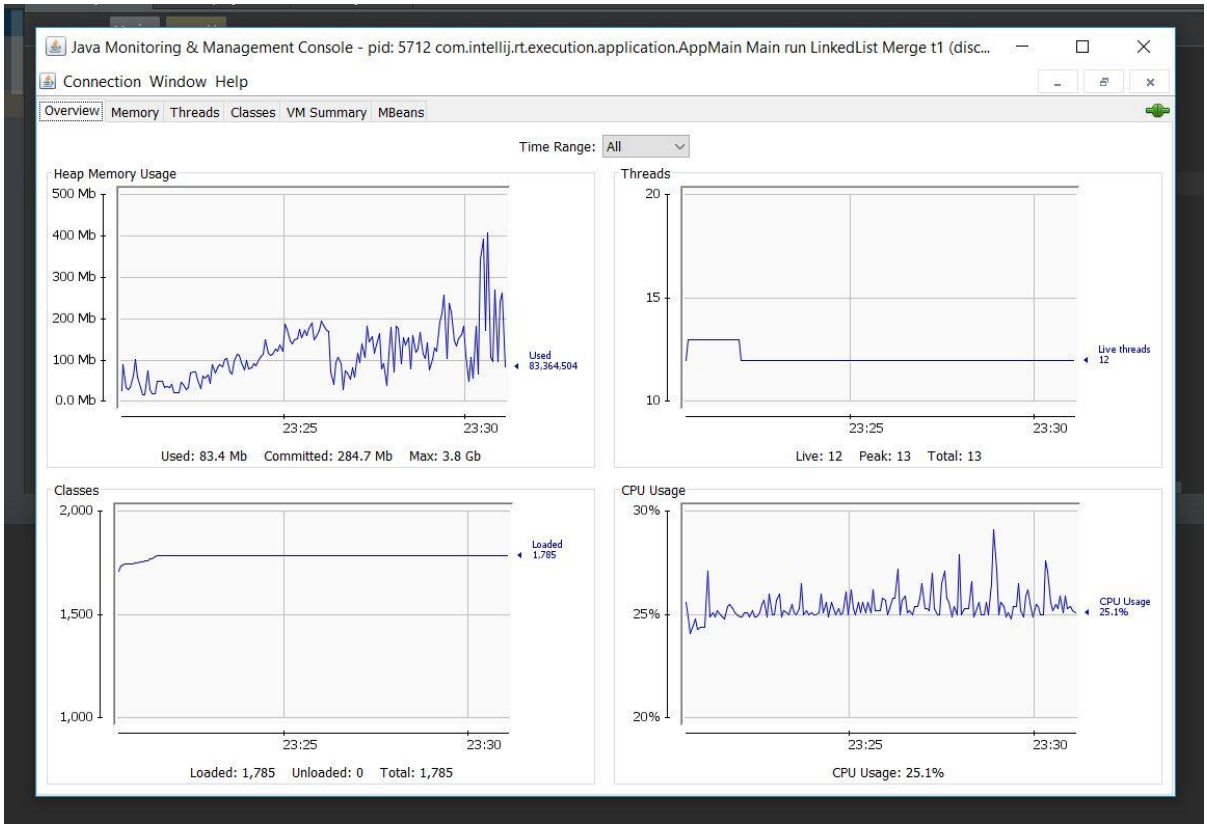
t1-Matrix-Merge

INFO - Time for creating graph in Matrix format: 203 mili seconds.

INFO - 5767 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 772 seconds.

INFO - Total time of program life cycle: 772 seconds.



با توجه به نمونه های مشاهده شده ، از نظر زمانی ذخیره در ساختار داده LinkedList بهتر از ذخیره در Matrix عمل می کند اما حافظه بیشتری می گیرد.

۲- مقایسه کارایی زمانی و حافظهای الگوریتم روی دو سیستم:

9) RUN LinkedList Optimum Insertion *N* t1

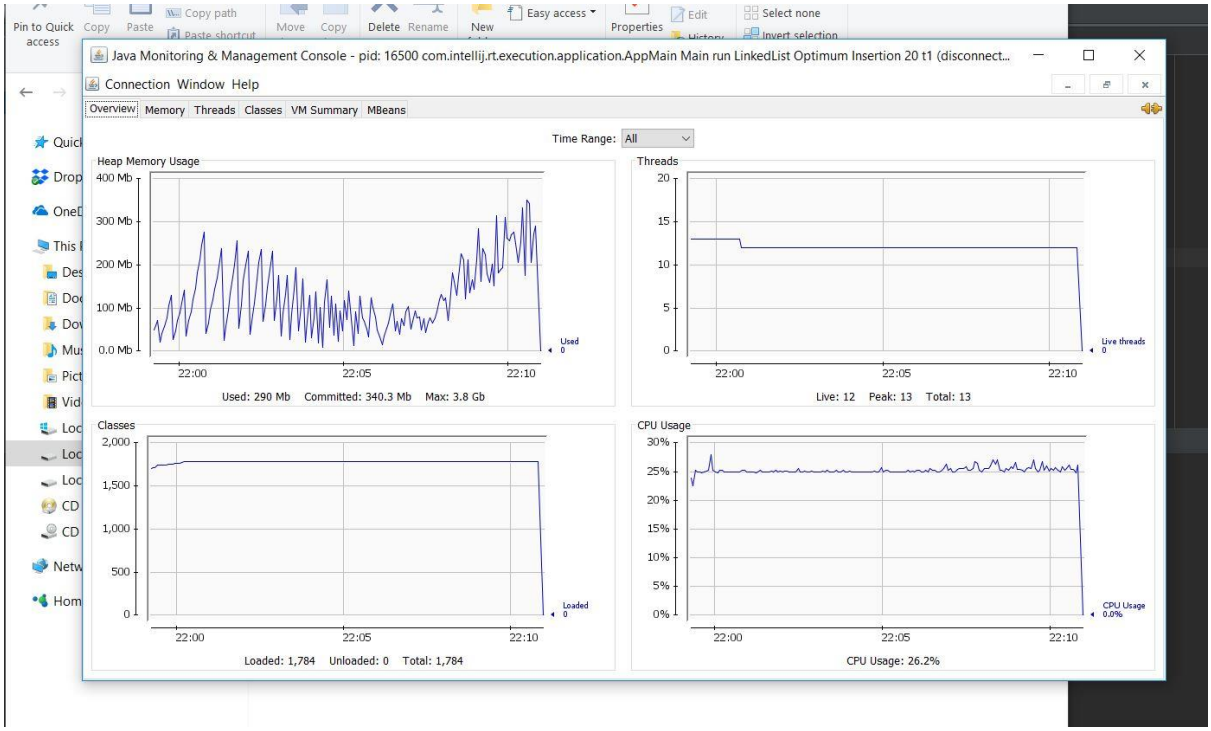
N = 20:

INFO - Time for creating graph in LinkedList format: 203 mili seconds.

INFO - 5555 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 11 minutes.

INFO - Total time of program life cycle: 11 minutes.



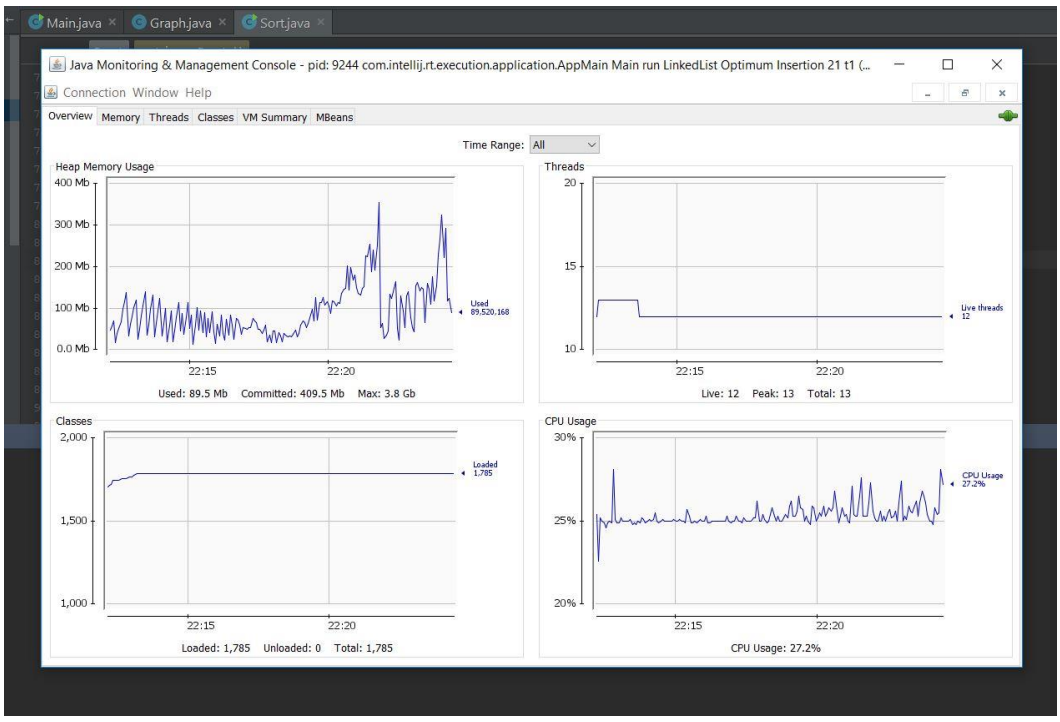
N = 21:

INFO - Time for creating graph in LinkedList format: 212 mili seconds.

INFO - 5837 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 12 minutes.

INFO - Total time of program life cycle: 12 minutes.



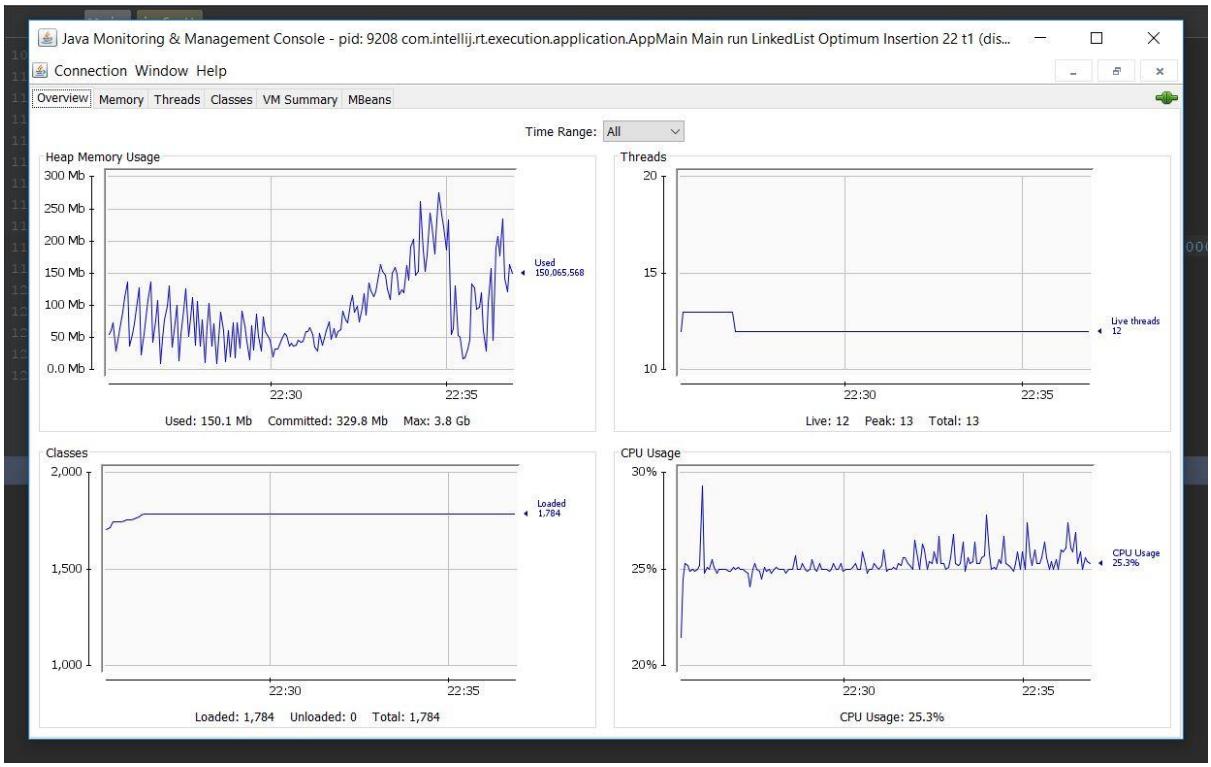
N = 22:

INFO - Time for creating graph in LinkedList format: 204 mili seconds.

INFO - 5619 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 11 minutes.

INFO - Total time of program life cycle: 11 minutes.



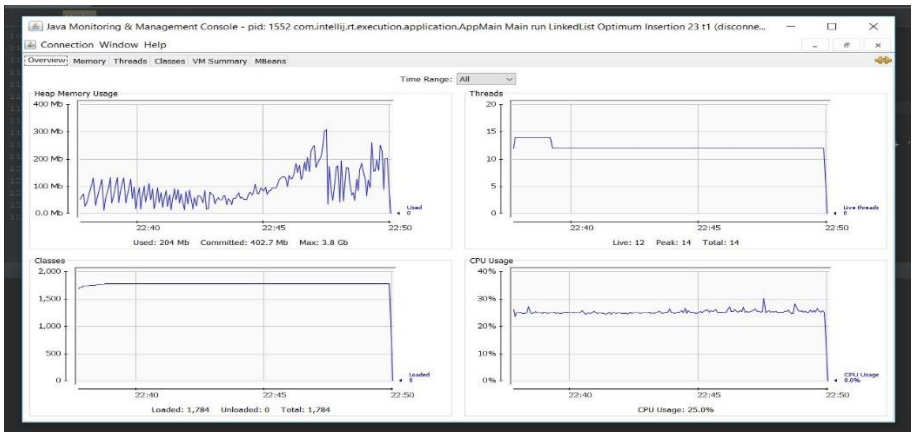
N = 23:

INFO - Time for creating graph in LinkedList format: 235 mili seconds.

INFO - 5679 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 12 minutes.

INFO - Total time of program life cycle: 12 minutes.



N = 24:

INFO - Time for creating graph in LinkedList format: 203 mili seconds.

INFO - 5465 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 11 minutes.

INFO - Total time of program life cycle: 11 minutes.

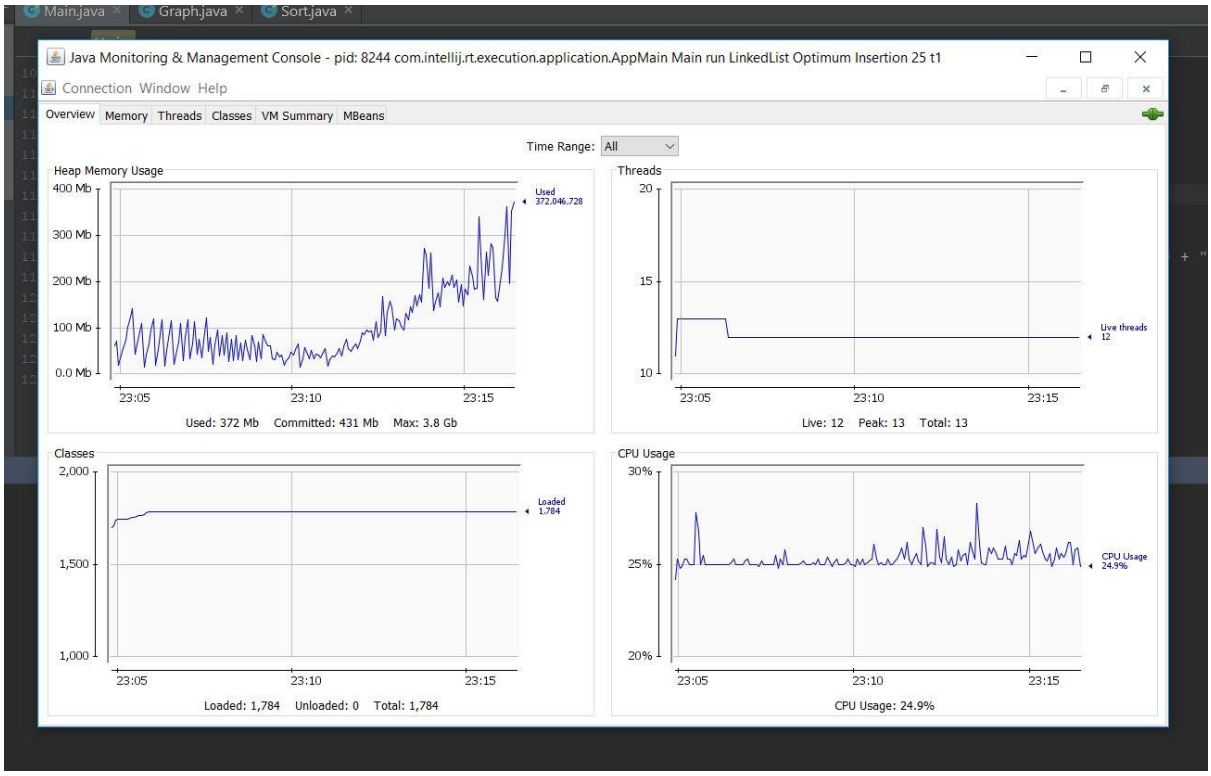
N = 25:

INFO - Time for creating graph in LinkedList format: 188 mili seconds.

INFO - 5591 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 11 minutes.

INFO - Total time of program life cycle: 11 minutes.



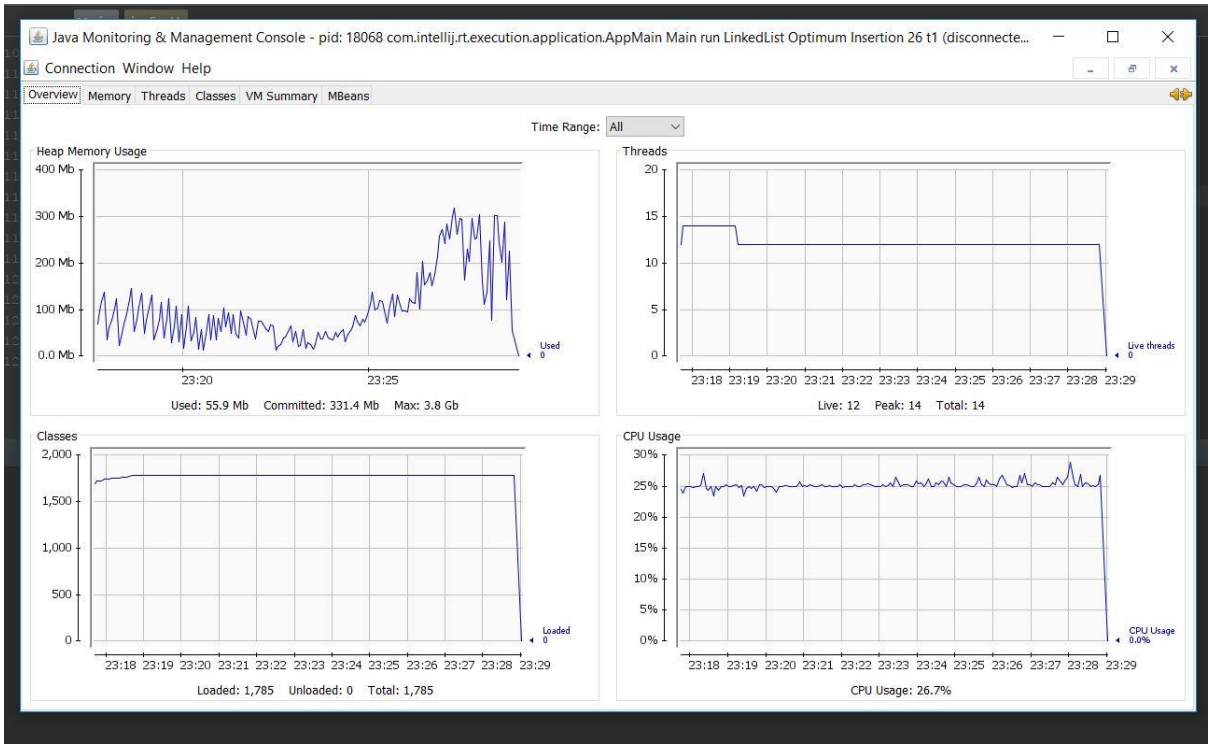
N = 26:

INFO - Time for creating graph in LinkedList format: 203 mili seconds.

INFO - 5507 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 11 minutes.

INFO - Total time of program life cycle: 11 minutes.



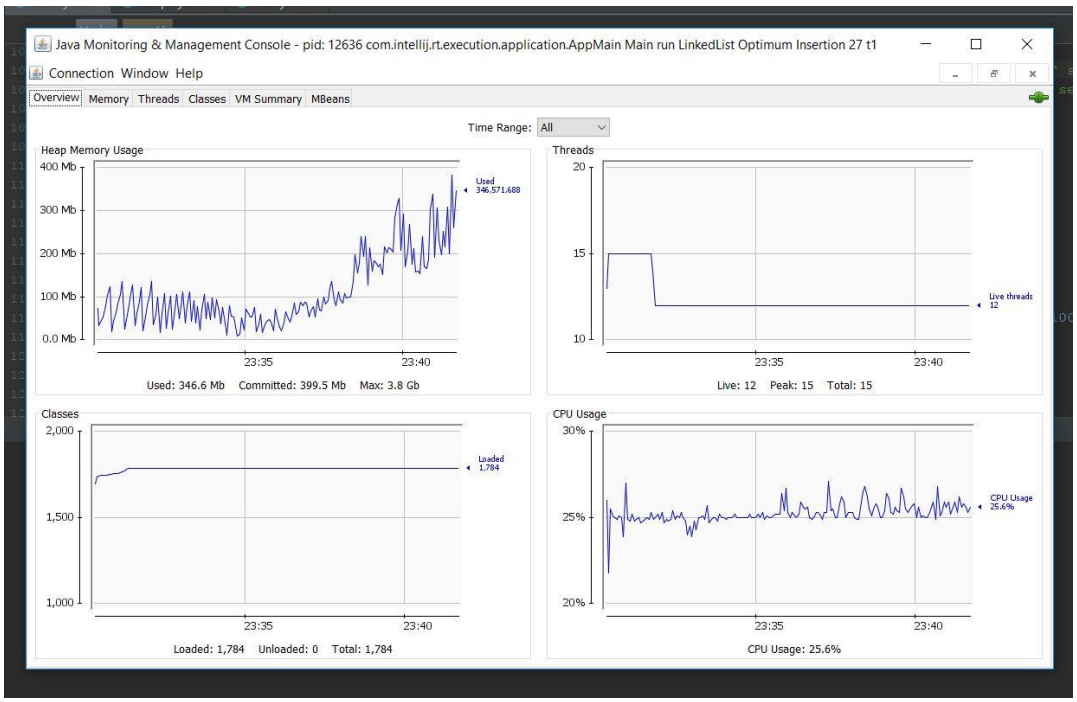
N = 27:

INFO - Time for creating graph in LinkedList format: 235 mili seconds.

INFO - 5558 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 11 minutes.

INFO - Total time of program life cycle: 11 minutes.



در این حالت با توجه به نمودارهای حافظه و زمان های به دست آمده ، مقدار ایده آل N برابر ۲۴ می شود .

10) RUN Matrix Optimum Insertion *N t2*

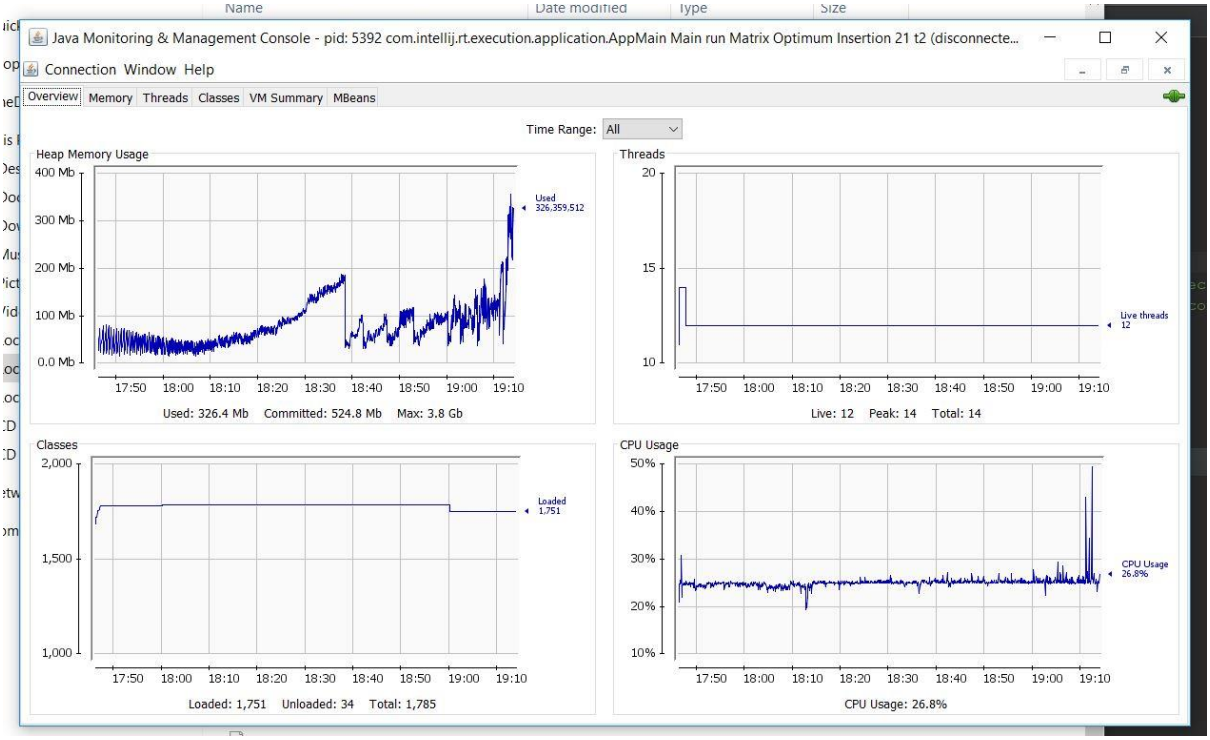
N = 21

INFO - Time for creating graph in Matrix format: 489 mili seconds.

INFO - 11242 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 5277 seconds.

INFO - Total time of program life cycle: 5277 seconds.



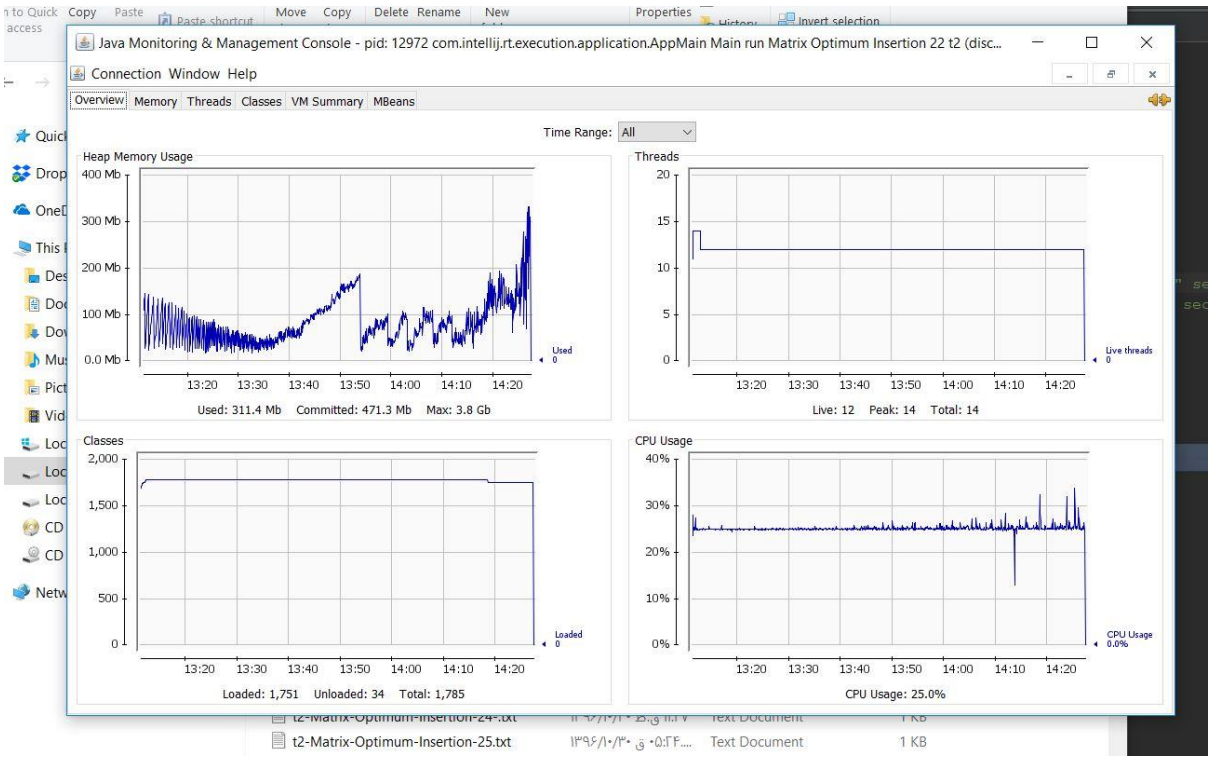
N = 22

INFO - Time for creating graph in Matrix format: 297 mili seconds.

INFO - 11090 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 4726 seconds.

INFO - Total time of program life cycle: 4726 seconds.



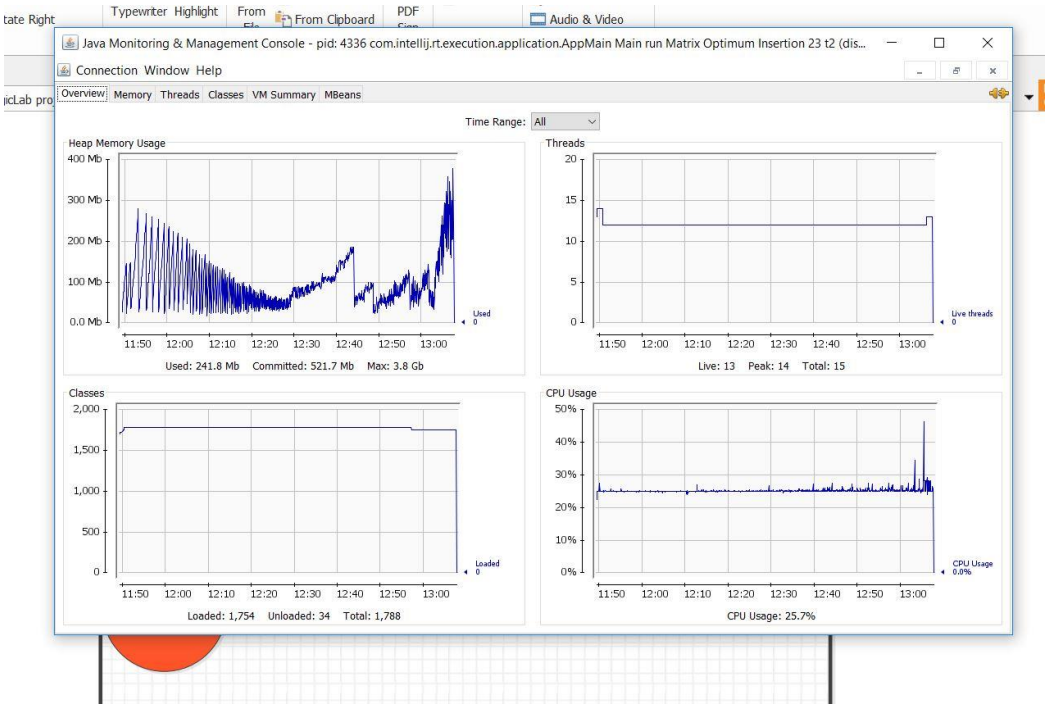
N = 23

INFO - Time for creating graph in Matrix format: 282 mili seconds.

INFO - 11198 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 4702 seconds.

INFO - Total time of program life cycle: 4702 seconds.



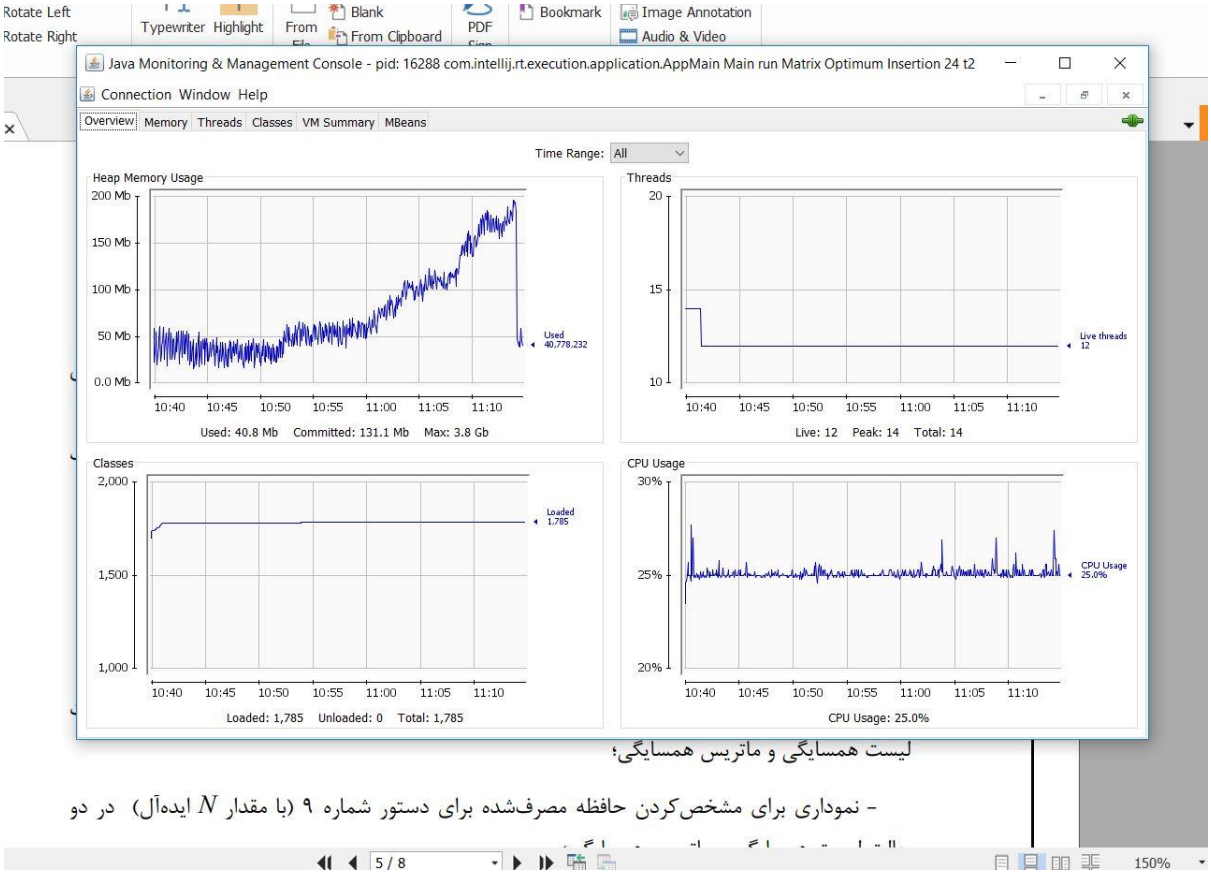
N = 24

INFO - Time for creating graph in Matrix format: 625 mili seconds.

INFO - 10963 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 4580 seconds.

INFO - Total time of program life cycle: 4581 seconds.



N = 25

INFO - Time for creating graph in Matrix format: 270 mili seconds.

INFO - 11190 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 4885 seconds.

INFO - Total time of program life cycle: 4885 seconds.

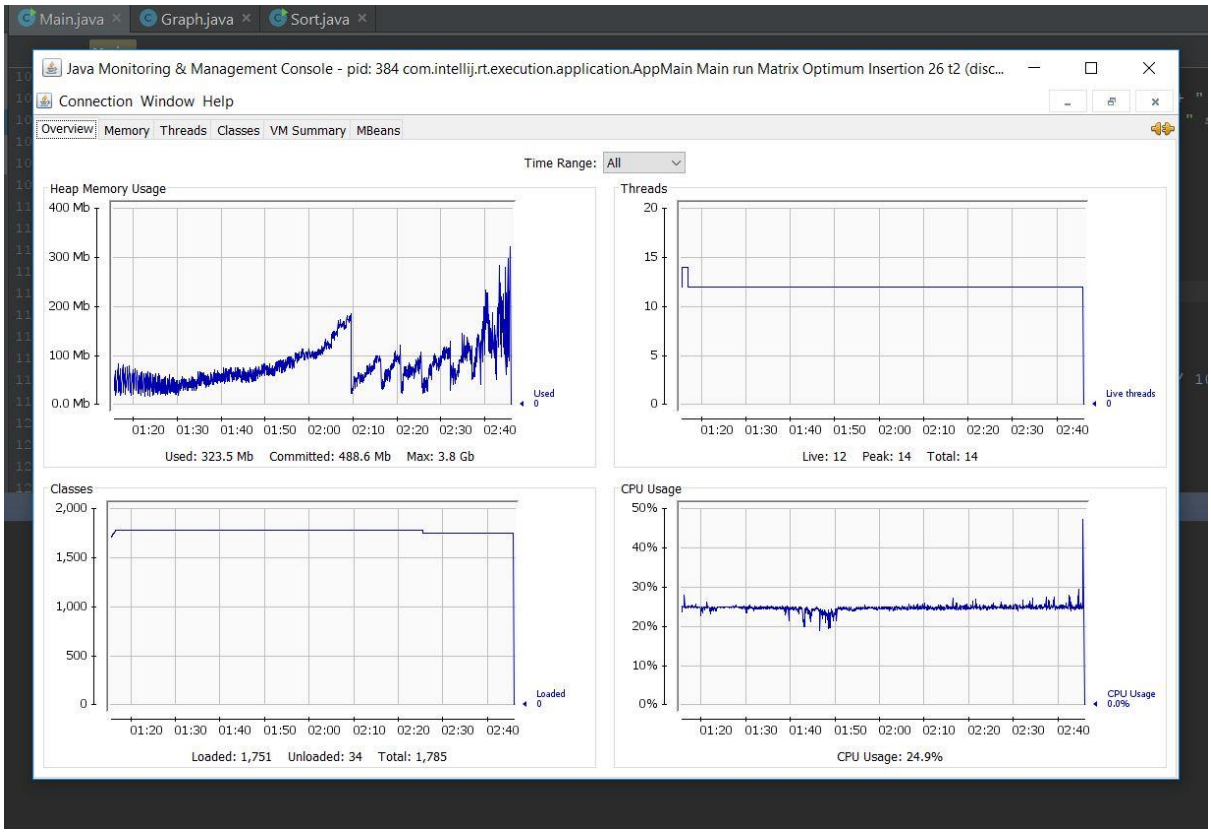
N = 26

INFO - Time for creating graph in Matrix format: 250 mili seconds.

INFO - 11167 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 5424 seconds.

INFO - Total time of program life cycle: 5424 seconds.



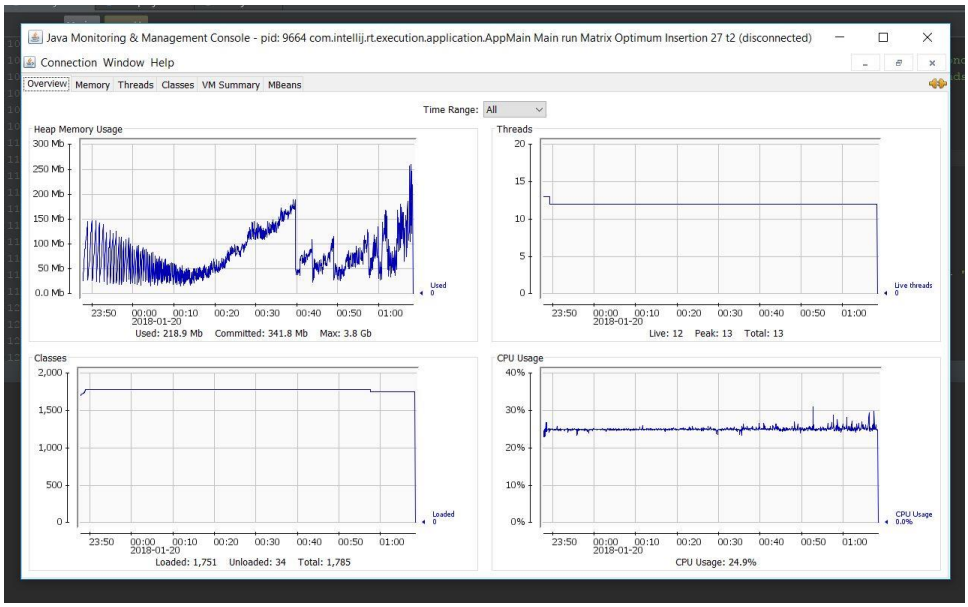
N = 27

INFO - Time for creating graph in Matrix format: 281 mili seconds.

INFO - 11088 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 4840 seconds.

INFO - Total time of program life cycle: 4841 seconds.



در این حالت با توجه به نمودارهای حافظه و زمان های به دســـت آمده ، کمترین میزان زمان ۴۵۸۰ ثانیه و کمترین میزان حافظه ماکسیمم 200MB است که هر با N = 24 حاصل می شوند . پس N ایده آل برابر 24 است .

11) RUN LinkedList Optimum Bubble *N t1*

N = 20

N = 21

N = 22

N = 23

N = 24

N = 25

N = 26

N = 27

12) RUN Matrix Optimum Bubble *N t1*

N = 20

N = 21

N = 22

N = 23

N = 24

N = 25

N = 26

N = 27

نمودار مشخص کننده ی زمان صرف شده برای دستور شماره ۹ با مقدار N ایده آل در دو حالت لیست همسایگی و ماتریس همسایگی :

Ideal N = 24

t1-LinkedList-Optimum-Insertion-24

INFO - Time for creating graph in LinkedList format: 203 mili seconds.

INFO - 5465 Edges are being deleted with this case.

INFO - Total time of executing algorithm: 11 minutes.

INFO - Total time of program life cycle: 11 minutes.

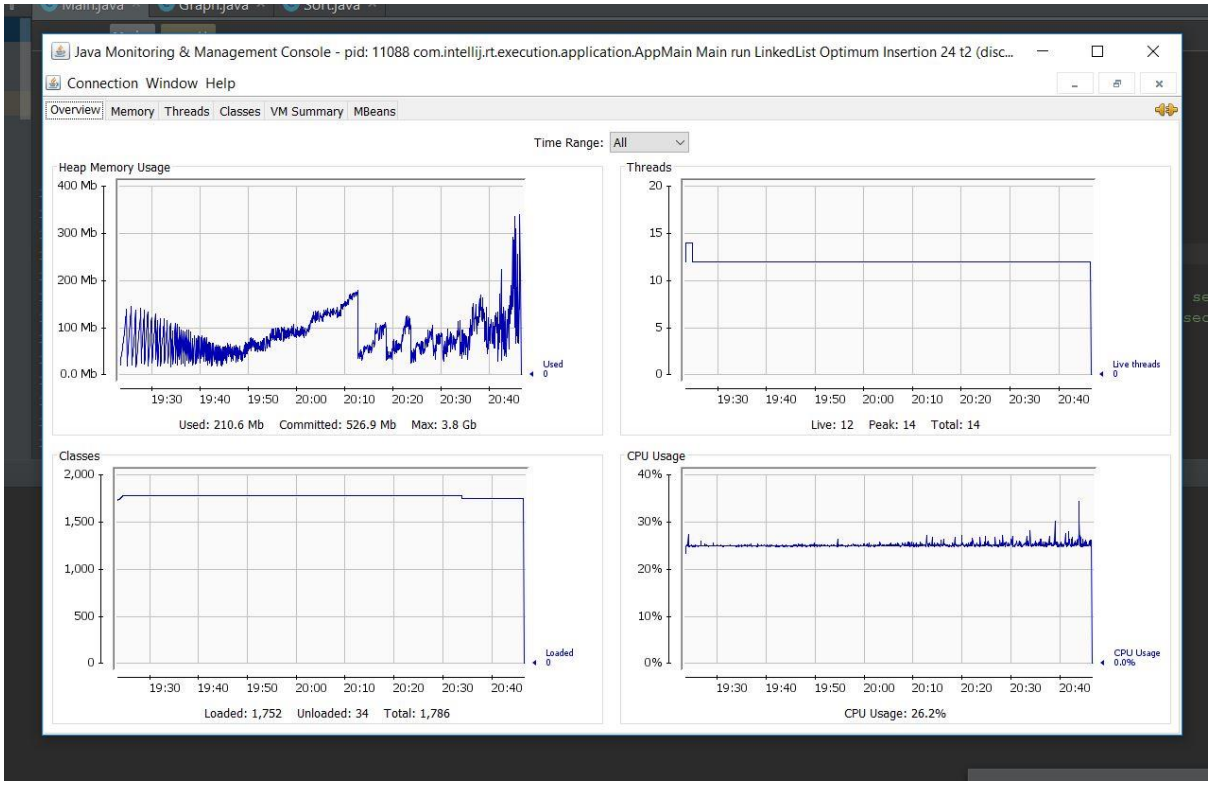
t2-LinkedList-Optimum-Insertion-24

INFO - Time for creating graph in LinkedList format : 266 mili seconds.

INFO - 11167 Edges are being deleted with this case.

INFO - Total time of executing algorithm : 4995 seconds.

INFO - Total time of program life cycle : 4995 seconds.



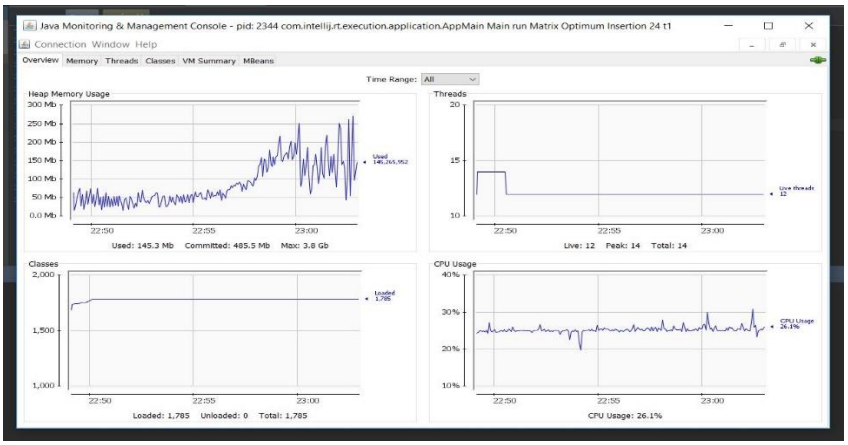
t1-Matrix-Optimum-Insertion-24

INFO - Time for creating graph in Matrix format : 219 mili seconds.

INFO - 5566 Edges are being deleted with this case.

INFO - Total time of executing algorithm : 841 seconds.

INFO - Total time of program life cycle : 841 seconds.



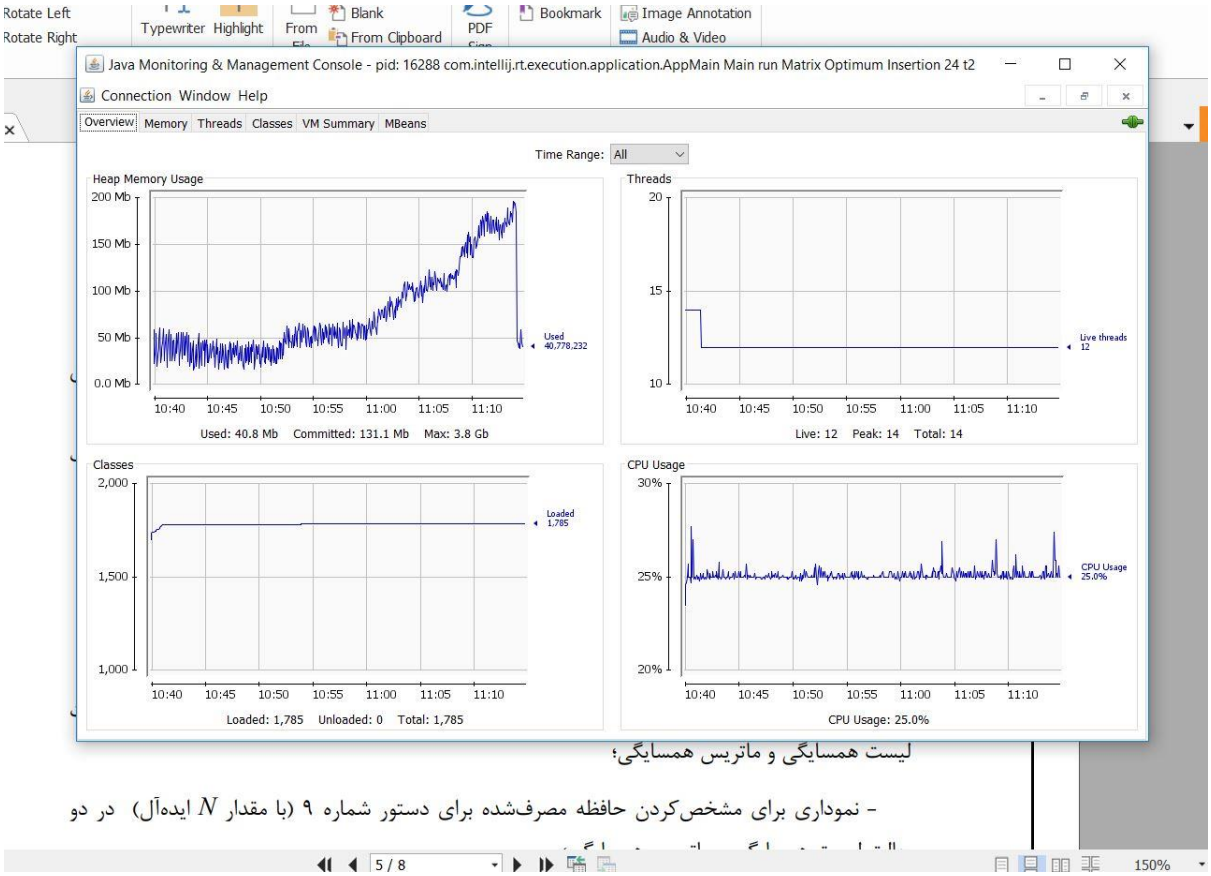
t2-Matrix-Optimum-Insertion-24

INFO - Time for creating graph in Matrix format : 625 mili seconds.

INFO - 10963 Edges are being deleted with this case.

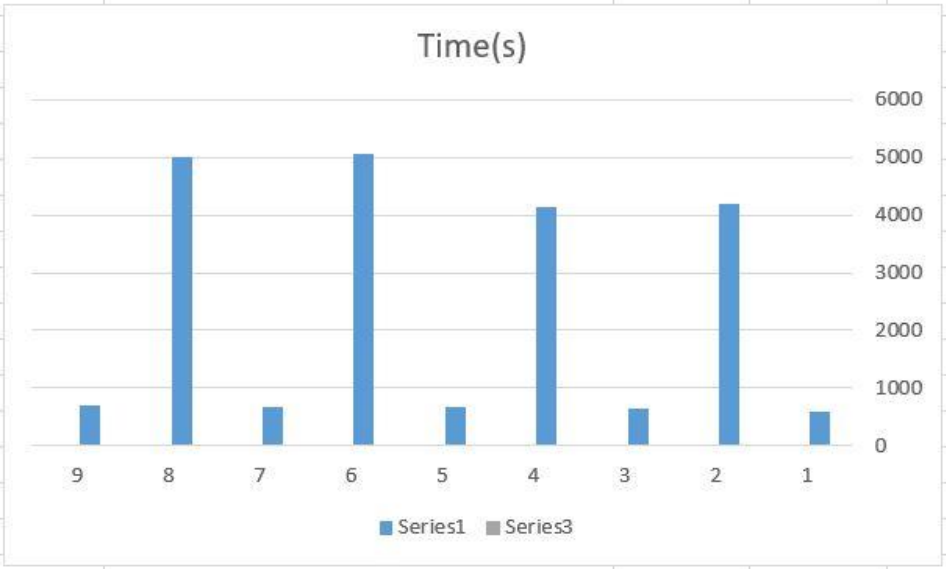
INFO - Total time of executing algorithm : 4580 seconds.

INFO - Total time of program life cycle : 4581 seconds.



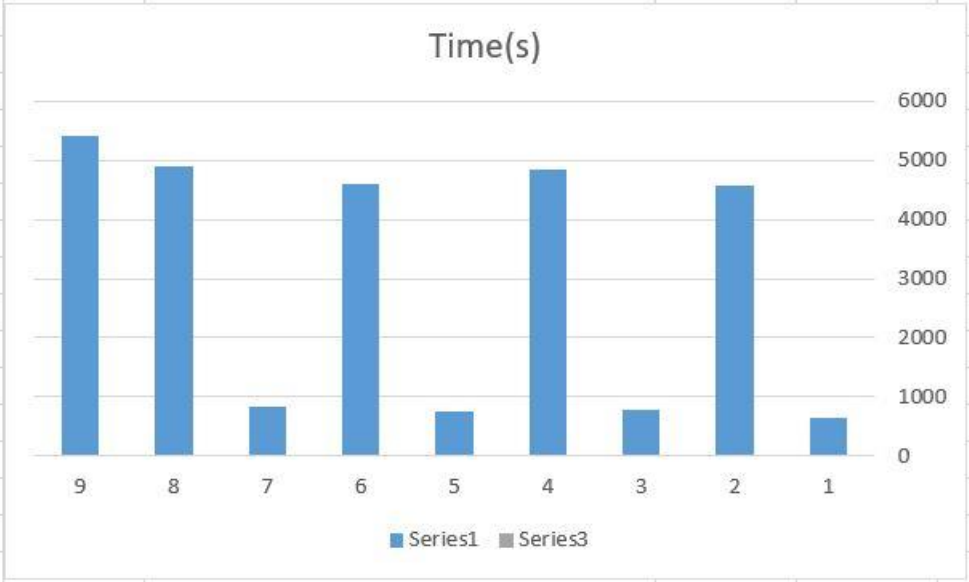
نموداری برای مقایسه زمان کل اجرای برنامه با همه د ستورات با شماره فرد (د ستورات مبتنی بر لیست همسایگی) روی دو تست کیس اول:

1	t1-LinkedList-Insertion	580
2	t2-LinkedList-Insertion	4200
3	t1-LinkedList-Merge	644
4	t2-LinkedList-Merge	4145
5	t1-LinkedList-Quick	660
6	t2-LinkedList-Quick	5073
7	t1-LinkedList-Optimum-Insertion-20	670
8	t2-LinkedList-Optimum-Insertion-24	4995
9	t1-LinkedList-Optimum-Insertion-27	700



1	t1-LinkedList-Insertion	340
2	t2-LinkedList-Insertion	460
3	t1-LinkedList-Merge	410
4	t2-LinkedList-Merge	510
5	t1-LinkedList-Quick	360
6	t2-LinkedList-Quick	190
7	t1-LinkedList-Optimum-Insertion-20	350
8	t2-LinkedList-Optimum-Insertion-24	340
9	t1-LinkedList-Optimum-Insertion-27	390

1	t1-Matrix-Insertion	642
2	t2-Matrix-Optimum-Insertion-24	4581
3	t1-Matrix-Merge	772
4	t2-Matrix-Optimum-Insertion-27	4840
5	t1-Matrix-Quick	740
6	t2-Matrix-Quick	4601
7	t1-Matrix-Optimum-Insertion-24	841
8	t2-Matrix-Optimum-Insertion-25	4885
9	t2-Matrix-Optimum-Insertion-26	5424



1	t1-Matrix-Insertion	480
2	t2-Matrix-Optimum-Insertion-24	190
3	t1-Matrix-Merge	470
4	t2-Matrix-Optimum-Insertion-27	260
5	t1-Matrix-Quick	380
6	t2-Matrix-Quick	210
7	t1-Matrix-Optimum-Insertion-24	270
9	t2-Matrix-Optimum-Insertion-26	320