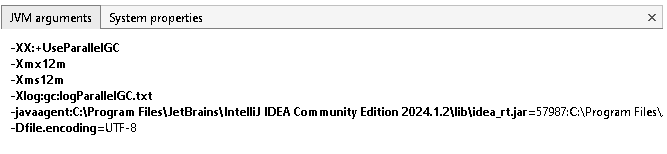
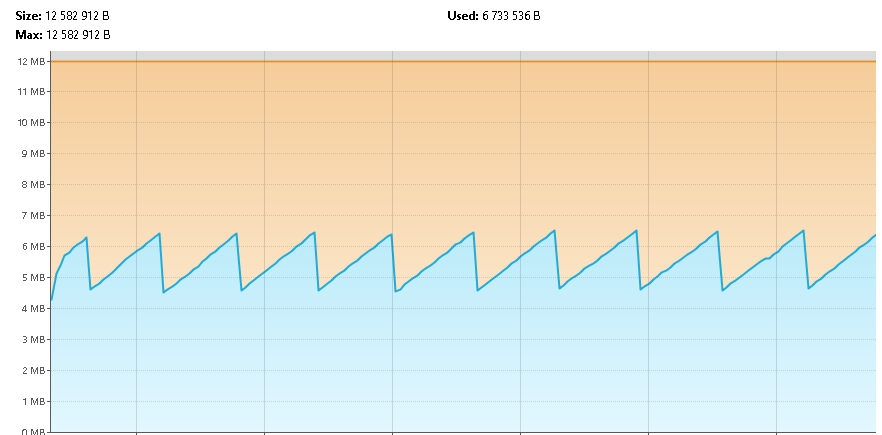
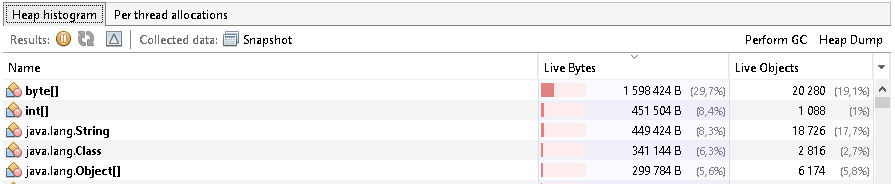
Запускаем программу. В режиме сборщика мусора Parallel.

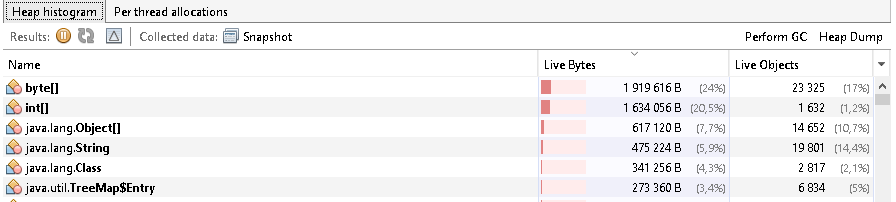
 Можно заметить, что при запуске в хип поступило почти 6MB объектов. Затем произошла малая сборка мусора и память упала до 4MB. Дальше можем наблюдать поднятие до 6 и снова сборку:



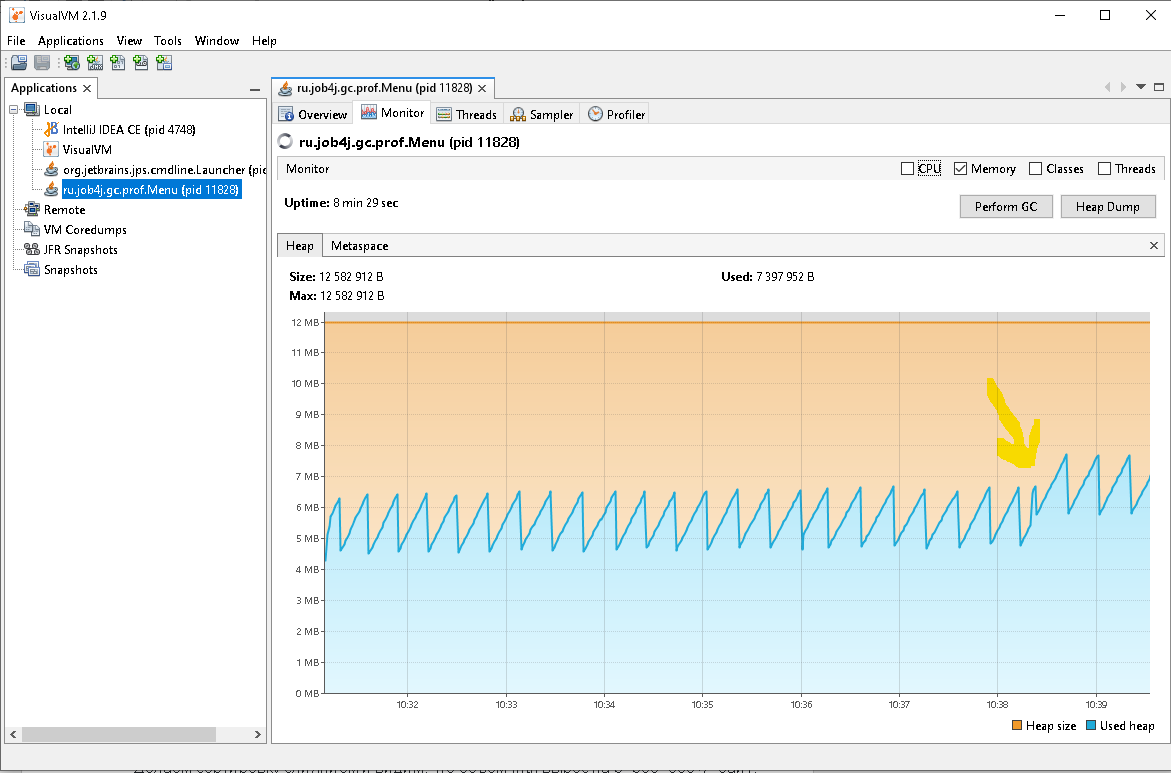
На вкладке **Sampler**, где можем отследить количество объектов каждого типа:



Далее создаем массив на 250000 элементов. Мы можем его заметить - int[] вырос на  размер массива (1\_000\_016).

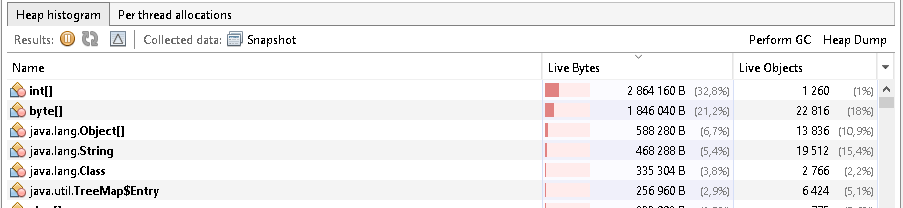


На графике этот процесс отобразился так:

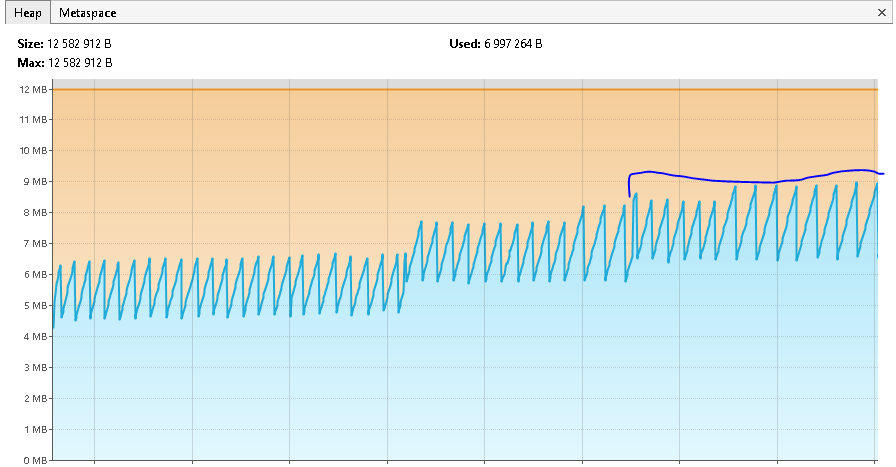


**Сортировка слиянием.**

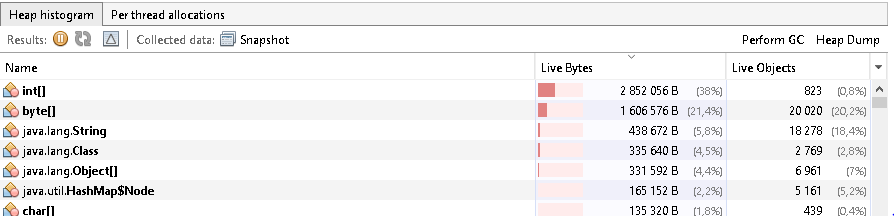
Делаем сортировку слиянием и видим, что объем int[] вырос на 2\_800\_000+/- байт. Учитываем, что мы также создали клон массива(1000000+ байт). И много памяти потребовалось при разбиении на массивы (почти 2300000+ байт). Вся сортировка заняла ~90 миллисекунд.



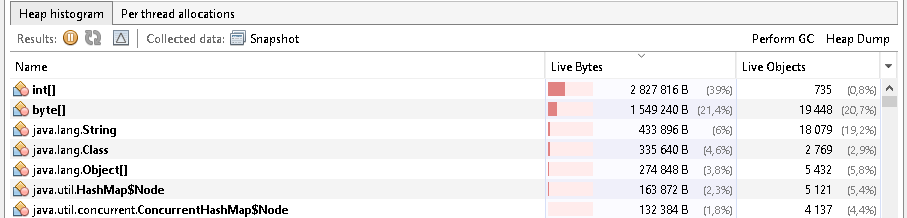
Этот резкий скачек видим на диаграмме (достигли 8,5 мегабайт):



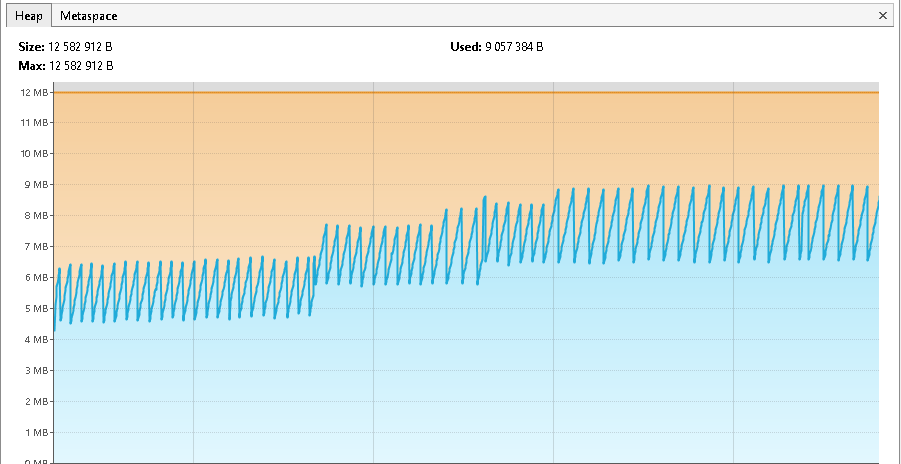
Затем было несколько малых сборок и хип теперь выглядит так:



**Сортировка методом вставки.**

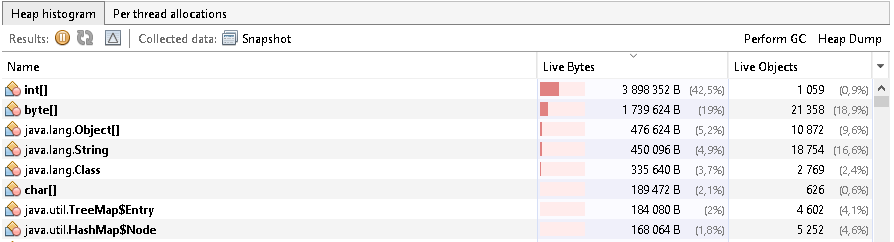
Теперь делаем сортировку методом вставки

По времени процедура заняла 7 секунд. Parallel GC пока обходится малыми сборками:

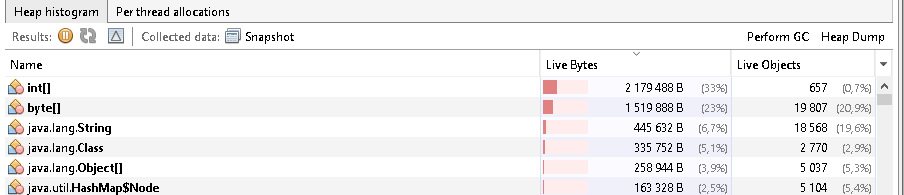


**Сортировка пузырьком.**

Начинаем сортировку пузырьком. В хип добавился клон массива:



Затем происходит полная сборка:



Сортировка пузырьком заняла 2 минуты. Здесь будет очень много перестановок элементов, но накладных расходов в хипе на данную операцию мы не наблюдаем.

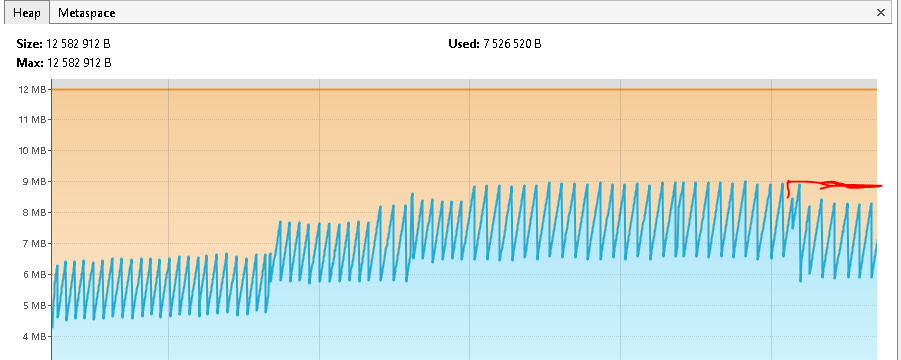
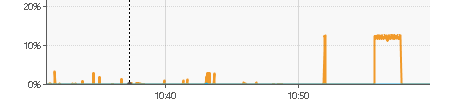
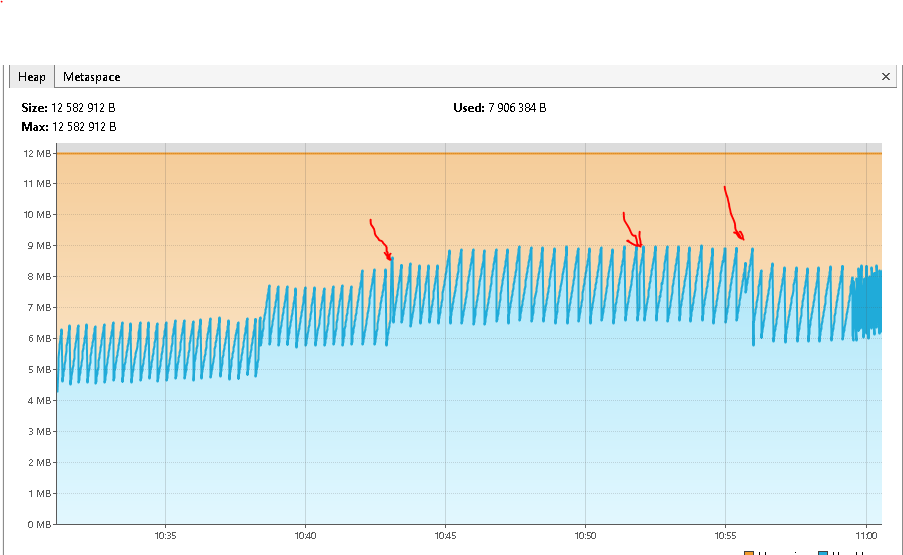


График загрузки процессора на каждой сборке. Сортировка пузырьком потребовала больше всего ресурсов процессора:



Общий график, на котором выделены сортировки:



Лог сборщика мусора показывает, что full сборка занимает 17.214ms

