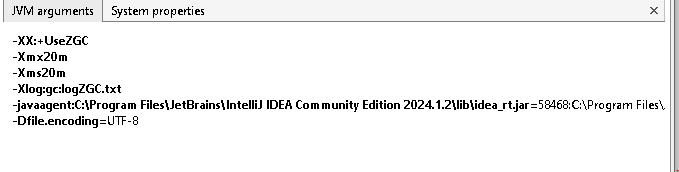
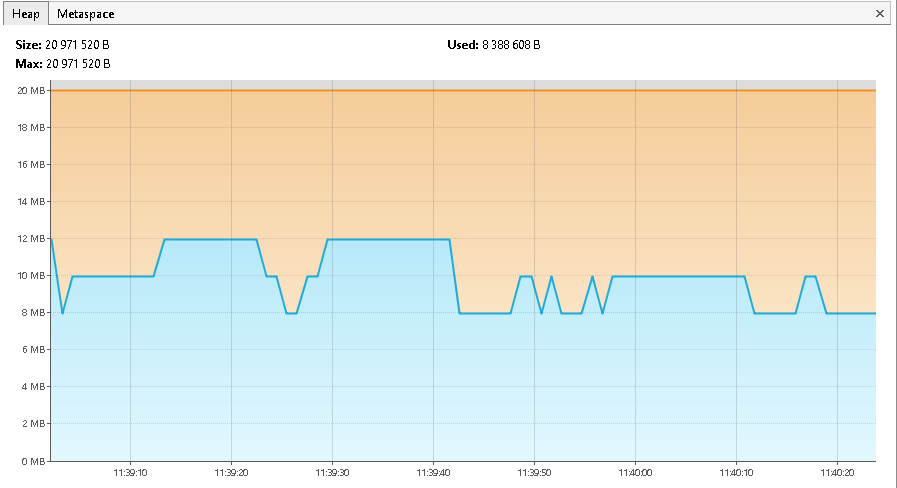
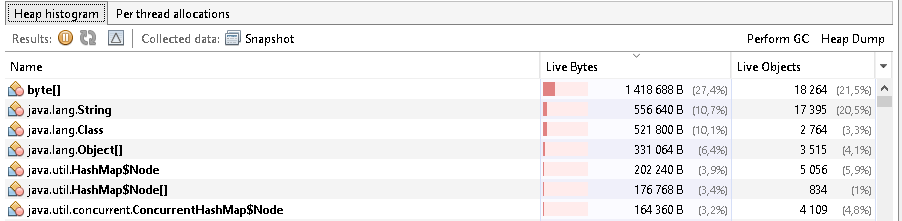
Запускаем программу. В режиме сборщика мусора ZGC с увеличением обьема heap до 20 MB.

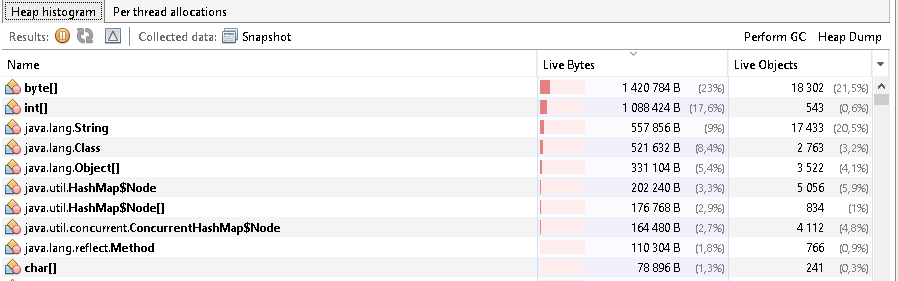
 Можно заметить, что при запуске в хип поступило 12MB объектов. Затем произошла малая сборка мусора и память упала до 8MB. Дальше можем наблюдать поднятие до 10 и снова сборку:



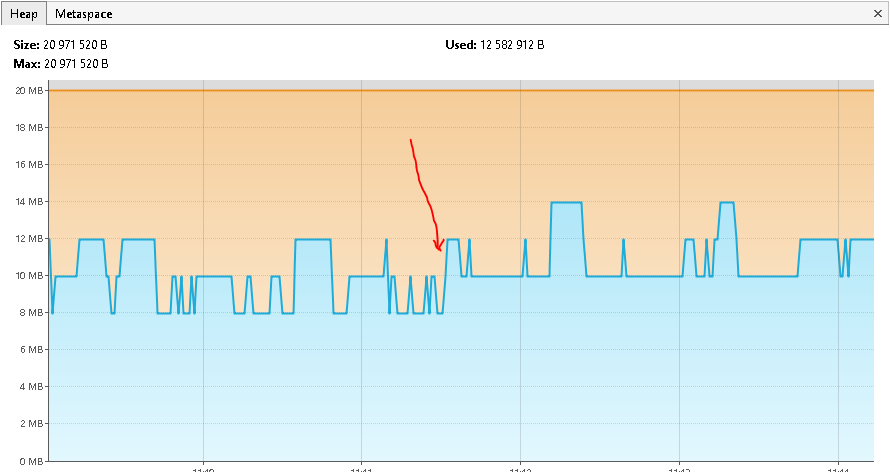
На вкладке **Sampler**, где можем отследить количество объектов каждого типа:



Далее создаем массив на 250000 элементов. Мы можем его заметить - int[] вырос на  размер массива (1\_000\_016) но затем произошла сборка мусора и часть объектов попала под нее.

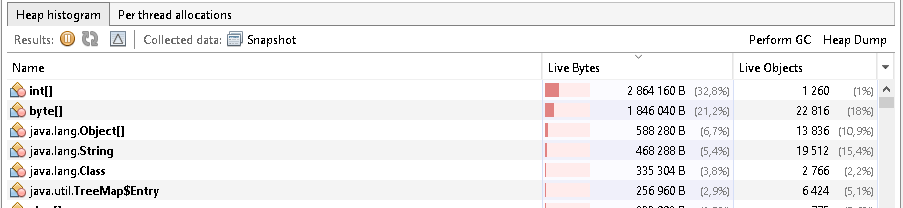


На графике этот процесс отобразился так:

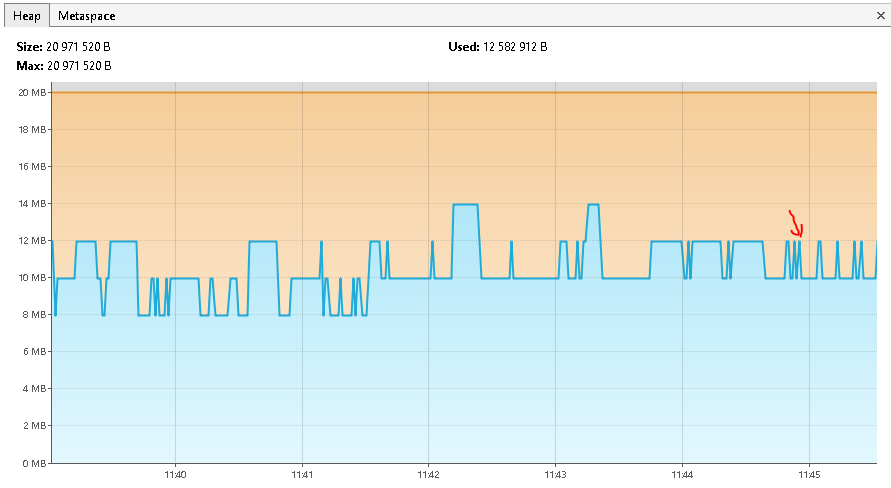


**Сортировка слиянием.**

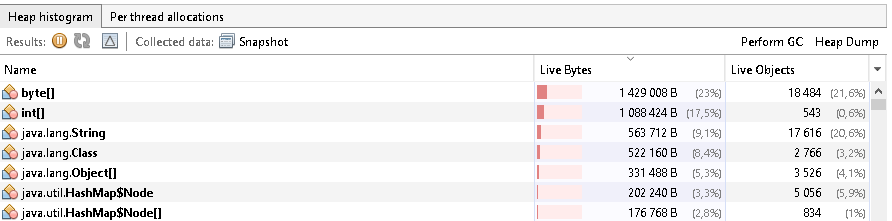
Делаем сортировку слиянием и видим, что объем int[] вырос на 2\_800\_000+/- байт. Учитываем, что мы также создали клон массива(1000000+ байт). И много памяти потребовалось при разбиении на массивы (почти 2300000+ байт). Вся сортировка заняла ~90 миллисекунд.



На увеличении объема памяти в этот раз операция не сказалась:

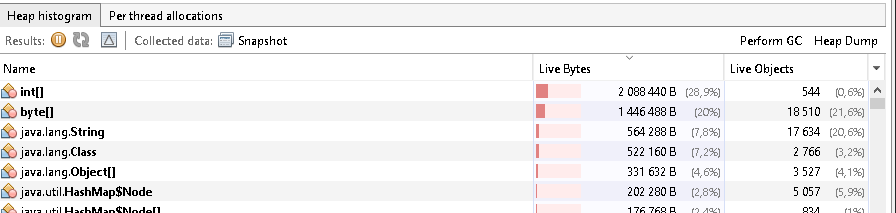


Затем было несколько малых сборок и хип теперь выглядит так:

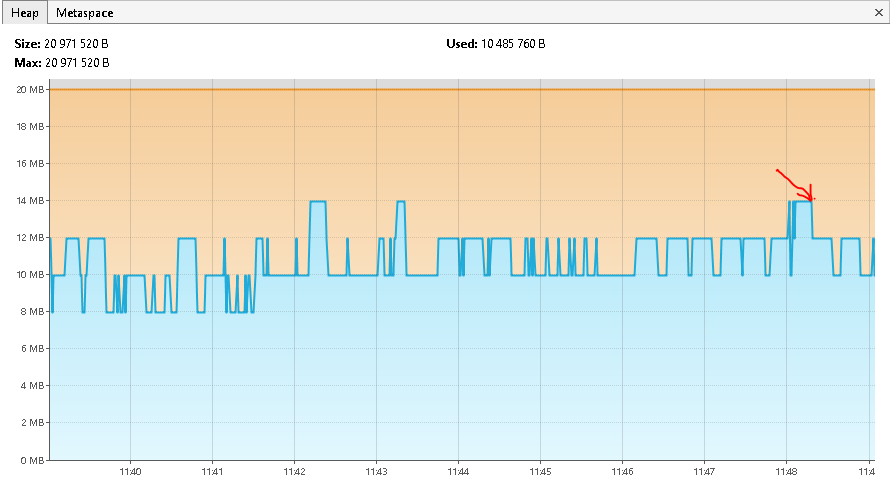


**Сортировка методом вставки.**

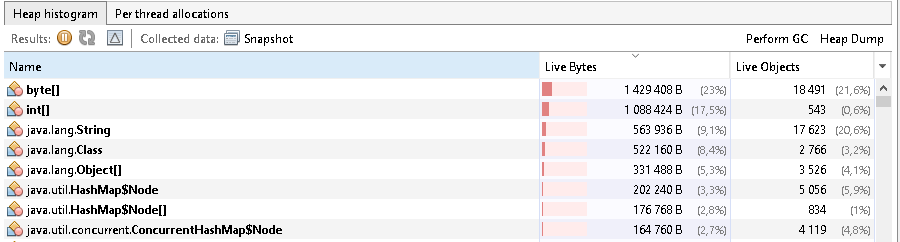
Теперь делаем сортировку методом вставками. Можем наблюдать, что хип вырос на размер массива, но существенных накладных расходов на данную операцию по сравнению с merge нет (мы не создаем массивы, а меняем ссылки у элементов (если это потребуется) + локальные переменные):



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

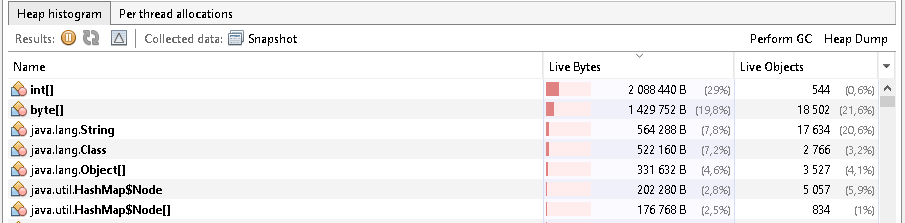


 Прошло несколько малых сборок и теперь объектов стало столько:

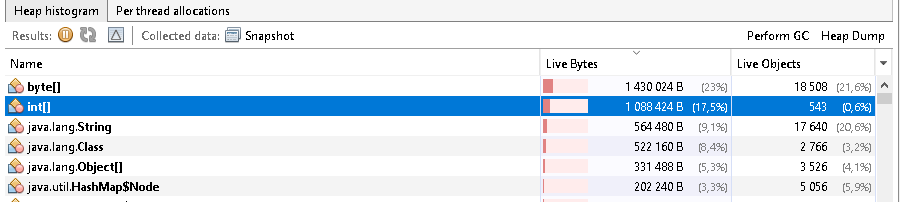


**Сортировка пузырьком.**

Начинаем сортировку пузырьком. В хип добавился клон массива:



Затем происходит полная сборка:



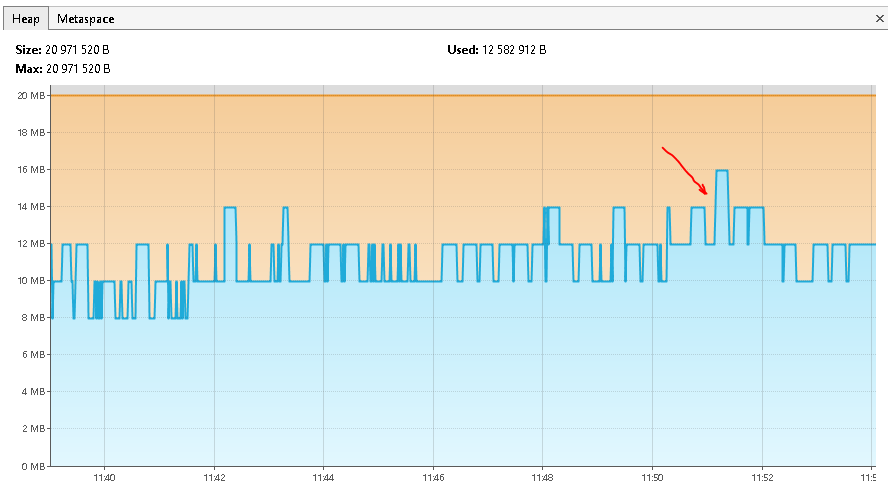
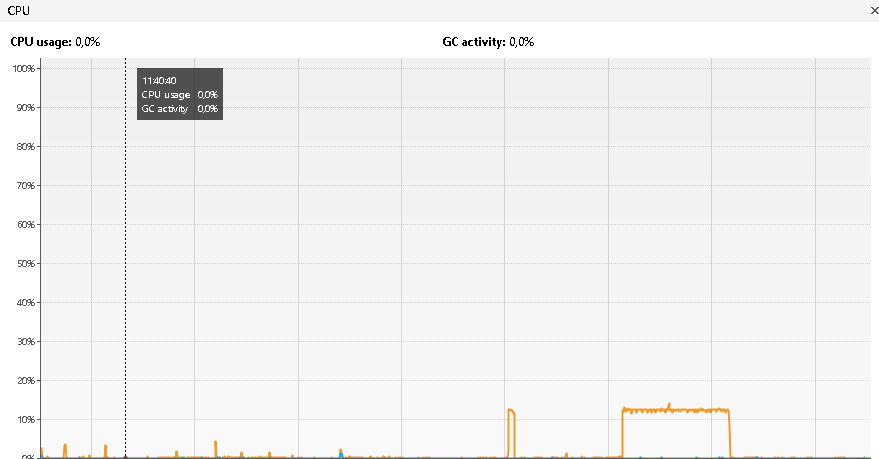
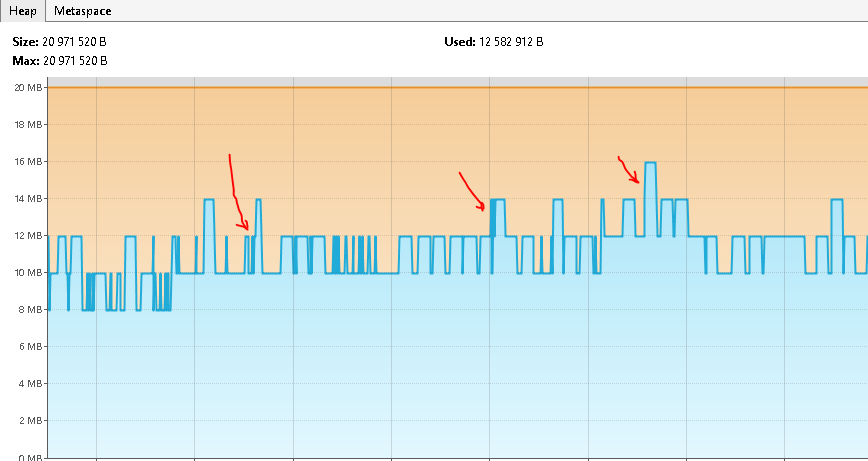
Сортировка пузырьком заняла 2 минуты 4 сек. Как видим – память ушла на создание клона массива и запустилась первая полная сборка. Здесь будет очень много перестановок элементов.

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



Общий график, на котором выделены сортировки:



Лог сборщика мусора logZGC.txt:

