Knapsack problem

1. This knapsack problem includes eleven different items. Each item has an id, benefit, and weight.
   a. The representation of the solution looks like this:
      $x = \{x_1, x_2,..,..,x_n\}$
      n = 11
   b. What we want to maximize in this problem, is the benefit. We always look after the item which has the best benefit.

      $$\sum b_i$$

   c. The knapsack problem is restricted to each items weight because the knapsack has a limit of how much weight it can carry.

      $$\sum w_i \leq W$$

2. In the knapsack-program, we are using two different algorithms, breadth-first, and depth-first, to find the best path.
   a. **Comparison of the time expended by the algorithms.**

Round	DFS	BFS
1	12 ms	6 ms
2	7 ms	6 ms
3	5 ms	5 ms
4	6 ms	5 ms
5	5 ms	5 ms
6	9 ms	7 ms

   As we see in the table above in almost all cases, *Breadth_First* executes faster than *Depth_First*. And in those cases, it is not faster, both algorithms have the same execution time. In the table above, it is two times out of 6 possible times that *BFS* and *DFS* have the same execution time, which makes *BFS* faster in 33 % of the cases.
   b. **Comparison of the space used in memory at a time by the algorithms.**

   The algorithms are using almost the same amount of memory, but *Depth_First* are using a very small amount more. *Breadth_First* are using 492,06 kb and *Depth_First* are using 492,09 kb. So *Depth_First* are only using 0,3 more kb.

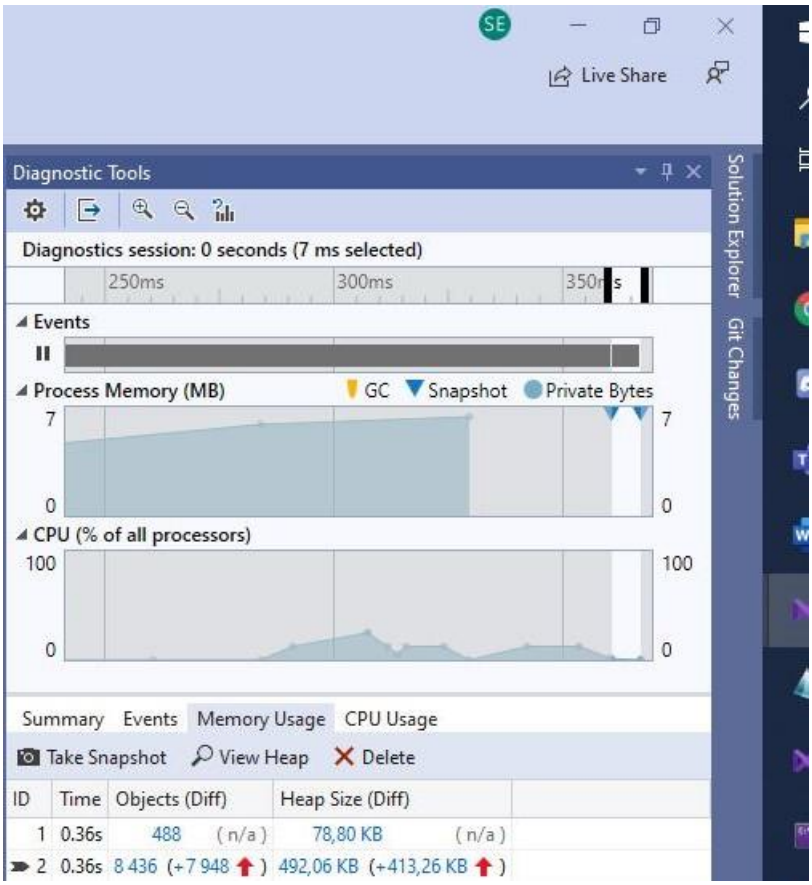

Figure 1: Breadth_First

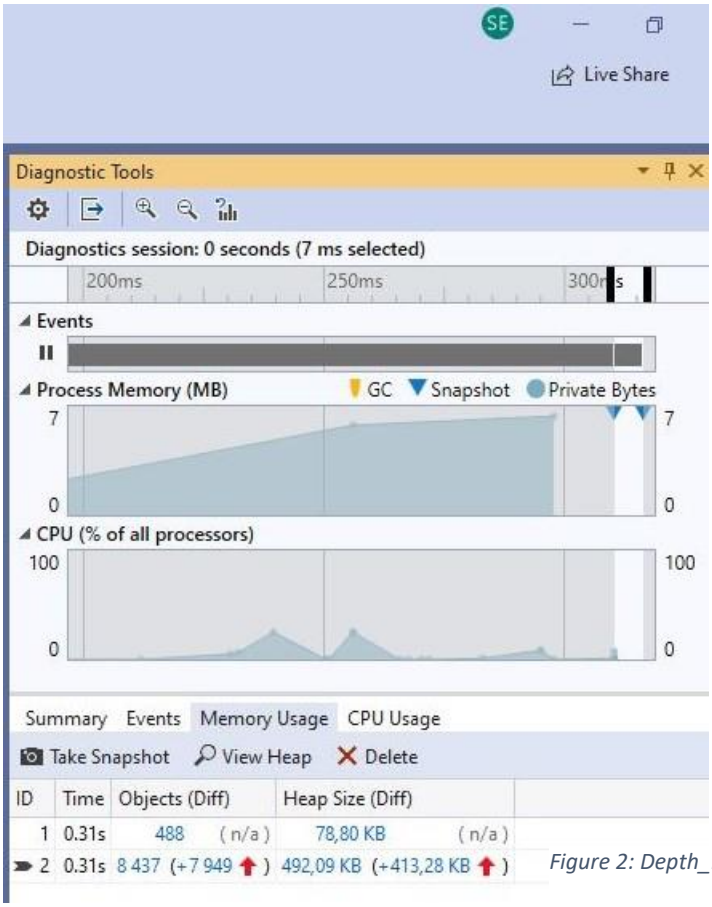

Figure 2: Depth_First