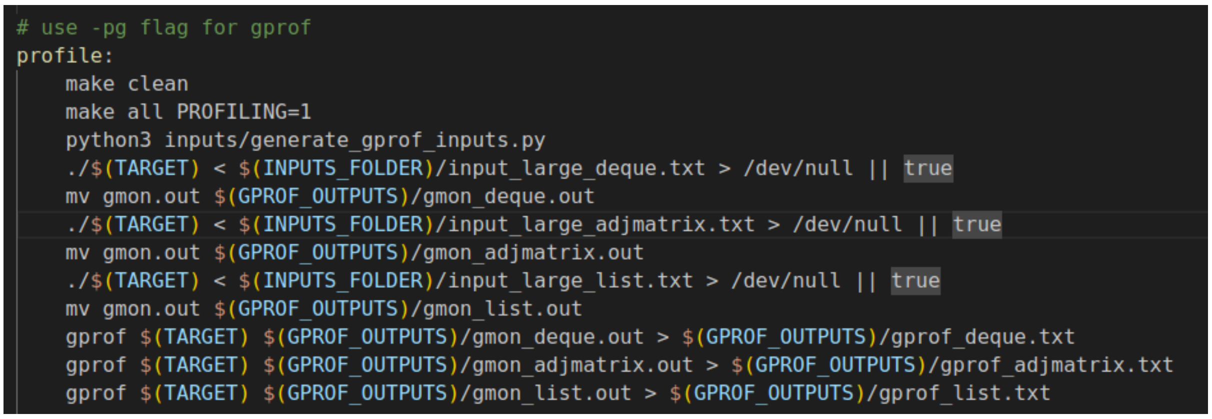
IDs: 53036281, 211791041

Question 2

We made a python script that creates large input files (10,000 vertices and 30,000 arches) to check which implementation is the fastest.
All input files are in the `/inputs` folder

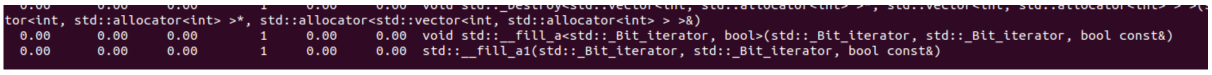
This is our `make profile`



```
# use -pg flag for gprof
profile:
	make clean
	make all PROFILING=1
	python3 inputs/generate_gprof_inputs.py
	./$(TARGET) < $(INPUTS_FOLDER)/input_large_deque.txt > /dev/null || true
	mv gmon.out $(GPROF_OUTPUTS)/gmon_deque.out
	./$(TARGET) < $(INPUTS_FOLDER)/input_large_adjmatrix.txt > /dev/null || true
	mv gmon.out $(GPROF_OUTPUTS)/gmon_adjmatrix.out
	./$(TARGET) < $(INPUTS_FOLDER)/input_large_list.txt > /dev/null || true
	mv gmon.out $(GPROF_OUTPUTS)/gmon_list.out
	gprof $(TARGET) $(GPROF_OUTPUTS)/gmon_deque.out > $(GPROF_OUTPUTS)/gprof_deque.txt
	gprof $(TARGET) $(GPROF_OUTPUTS)/gmon_adjmatrix.out > $(GPROF_OUTPUTS)/gprof_adjmatrix.txt
	gprof $(TARGET) $(GPROF_OUTPUTS)/gmon_list.out > $(GPROF_OUTPUTS)/gprof_list.txt
```

It recreates all objects with the -pg flag to enable profiling, creates all the input files using our python script and for each implementation we run the file with the large input and move the gmon output so it doesn't get overridden.
At the end we gprof all the outputs to a txt file to see easily
The last line of each functions list in the cumulative shows the total time it took for everything:

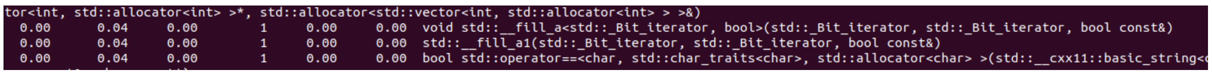
Deque (from `inputs/gprof_deque`):



```
  0.00      0.00     0.00        1     0.00     0.00  void std::_Destroy<std::vector<int, std::allocator<int> >*, std::vector<int, std::allocator<int> > >(s
tor<int, std::allocator<int> >*, std::allocator<std::vector<int, std::allocator<int> > >&)
  0.00      0.00     0.00        1     0.00     0.00  void std::__fill_a<std::_Bit_iterator, bool>(std::_Bit_iterator, std::_Bit_iterator, bool const&)
  0.00      0.00     0.00        1     0.00     0.00  std::__fill_a1(std::_Bit_iterator, std::_Bit_iterator, bool const&)
```

We have **0.00** seconds cumulative time

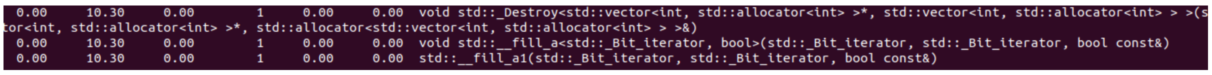
List (from `inputs/gprof_list`):



```
tor<int, std::allocator<int> >*, std::allocator<std::vector<int, std::allocator<int> > >&)
  0.00      0.04     0.00        1     0.00     0.00  void std::__fill_a<std::_Bit_iterator, bool>(std::_Bit_iterator, std::_Bit_iterator, bool const&)
  0.00      0.04     0.00        1     0.00     0.00  std::__fill_a1(std::_Bit_iterator, std::_Bit_iterator, bool const&)
  0.00      0.04     0.00        1     0.00     0.00  bool std::operator==<char, std::char_traits<char>, std::allocator<char> >(std::__cxx11::basic_string<
```

Cumulative time is very low but higher than Deque's: **0.04** seconds

AdjMatrix (from `inputs/gprof/adjmatrix`):



```
 0.00     10.30     0.00        1     0.00     0.00  void std::_Destroy<std::vector<int, std::allocator<int> >*, std::vector<int, std::allocator<int> > >(s
tor<int, std::allocator<int> >*, std::allocator<std::vector<int, std::allocator<int> > >&)
 0.00     10.30     0.00        1     0.00     0.00  void std::__fill_a<std::_Bit_iterator, bool>(std::_Bit_iterator, std::_Bit_iterator, bool const&)
 0.00     10.30     0.00        1     0.00     0.00  std::__fill_a1(std::_Bit_iterator, std::_Bit_iterator, bool const&)
```

This is the worst one with **10.30** seconds

To make sure deque is really the fastest out of those 3, we tested again with 100,000 and 300,000, which got us: deque with 0.14 seconds and list with 0.17. Not a big difference but from now on we will use Deque because it's the fastest implementation.