

(d)

```
File Edit Selection Find View Goto Tools Project Preferences Help
pointer.c — Lab\Lab1\lab1a x pointer.c — Lab sIn\lab1a x test.c x
38 ##### d #####
39 no_pre_alloc_start = time.perf_counter()
40 longlongstring = []
41 for i in range(t):
42     for line in longstring:
43         longlongstring.append(line)
44 no_pre_alloc_end = time.perf_counter()
45
46 no_pre_alloc_runtime = no_pre_alloc_end - no_pre_alloc_start
47 print("(d)Non-preallocate runtime: ", no_pre_alloc_runtime)
48
49 length = len(longlongstring)
50
```

This is a nested for loop, which will take $O(t*n)$ times. Also, without preallocation of the size of list, when we read a new string and want to put the value inside the list. This will force the memory to reallocate for every iteration, meaning that we need another $O(n)$ time to allocate the memory. And the total run time will be bounded by $O(n^2)$

(e)

```
50
51 ##### e #####
52 pre_alloc_start = time.perf_counter()
53 dt = np.dtype('<U' + str(k))
54 arr = np.empty(len_longstr, dtype = dt)
55
56 for j in range(len(longstring)):
57     arr[j] = longstring[j]
58 arr = np.tile(arr, t) # copy arr t times and save it as arr.
59 pre_alloc_end = time.perf_counter()
60 # print(arr)
61 pre_alloc_runtime = pre_alloc_end - pre_alloc_start
62 print("(e)Preallocate runtime: ", pre_alloc_runtime)
63
```

In this case, since we have already know the length of the list. We can use numpy.empty to create an empty array, meaning that we have already assigned a memory space for the value to store. Therefore, whenever we have a new value, we only need $O(1)$ to allocate the value into the array. When I form the array for the first time, I used tile() function to expand the array 100 times. This will create a exactly array of the previous one, except the data type it store is a numpy array. The total run time of this implementation will be $O(n)$ since I only have to run through the list for a first time.

(f)

```
(d)Non-allocate runtime: 0.008652793000000103  
(e)Preallocate runtime: 0.0005193949999999781  
[Finished in 0.7s]
```

Clearly, we can see that the runtime is faster when I preallocate it .