HW1 Part 2: Explore compromise_search.h

Wenyi Liu

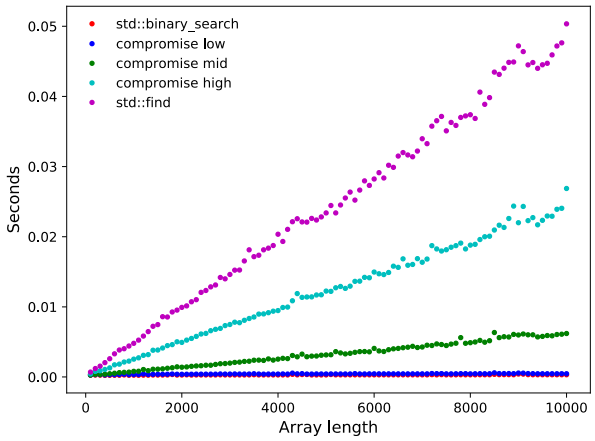
Q: What's the big-O of std::find?
A: std::find takes O(N) time because it is a linear search.

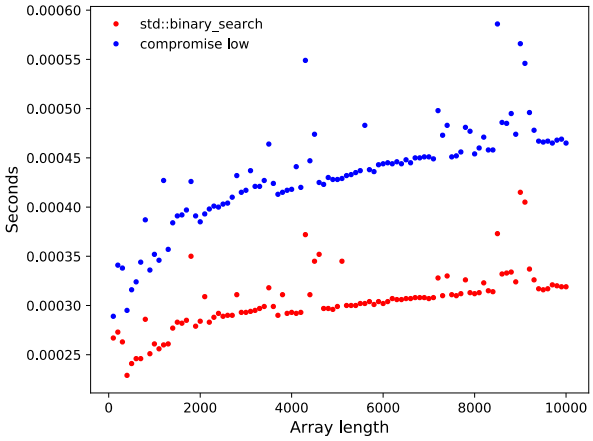
Q: What is the big-O of std::binary_search?
A: std::binary_search takes O(log N) time. In binary search, we narrow down the search space by half. The search space reduces to 0 after $1+\log_2 N$ times and the while loop terminates.

Q: What is the big-O of your code?
A: Without loss of generality, let N be the size of the initial search space, and n be the small_size. Then we need to do binary search until the search space reduce to (N - n). This part takes O(log(N-n)) time. The remaining linear search takes O(n) time. So the overall time complexity of my code is O(log(N-n) + n).

Depending on the input, our code will behave differently.
If N >> n, then the code takes O(log N) time.
If n is comparable to N, then the code takes O(N) time.

The following two questions are based on the following figures. In the test_code_performance.cpp, I profile the running time of my code, std::find and std::binary_search. Compromise low, mid, high correspond to different small_size, with low being a fixed value of 10, mid being 20% of N, and high being 80% of N. And I use performance.py to make these two figures.



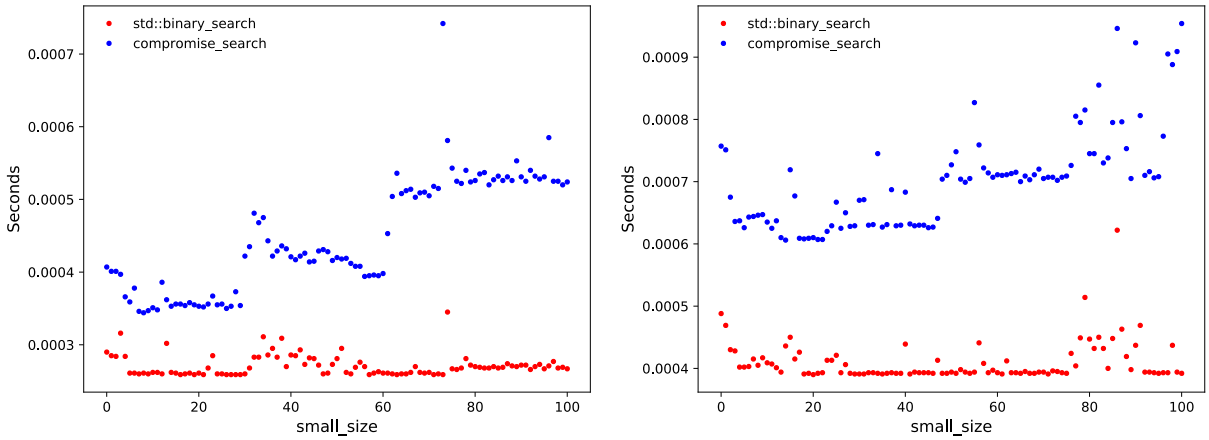


Q: On your computer in practice, does your code appear to have the big-O you expect?
A: From above, the code should take O(log(N-n) + n) time. So it should be slower than O(log N) but faster than O(N), which can

be seen from the left panel. The compromise search algorithm runs slower than std::binary_search but faster than std::find. When small_size is fixed at a very low value of 10, the compromise_search is slightly slower than binary search. In this regime, N >> n and compromise search should behave as O(log N), as can be seen from the right panel. When small_size is mid or high, i.e. n is comparable to N, the run time becomes O(N), but still faster than std::find. Thus, the code has the expected big-O.

Q: How fast is your code compared to std::binary_search in practice on your computer?
A: From the right panel of the figure, compromised search is slightly slower than std::binary search when the small_size is set to a fixed small constant. But both take O(log N) time.

The following question uses the following figure. In the choose_small_size.cpp, I profile the running time of my code and std::binary search v.s. small_size. I use small_size.py to make these two figures. Left panel has a search space of 1,000 and right panel has a search space of 100,000.



Q: What's a good value for small_size?
A: A good choice of small_size seems to consistently fall into the range between 10-30 regardless of the size of search space. At this size, std::find is as fast as std::binarysearch.