**표준 알고리즘 자료**

회사의 시험을 준비하는데 있어 습득해야 하는 거의 필수적인 핵심 알고리즘 모음이 있습니다. 회사에서는 이런 목록에 들어있지 않은 질문은 하지 않으려고 합니다. 좋은 프로그래머 다수가 이 목록에 없는 질문에 대해서는 답을 모르기 때문이며 그래서 목록 외 질문을 냈다가 회사는 좋은 사람을 뽑는데 실패하게 됩니다.

알아야 할 자료구조는 다음과 같습니다.

* list 구조: 배열, 동적 배열, 링크드 리스트(linked list)
* set과 map 구조: 해시맵, 이진 검색 트리, 힙

여기서 언급한 자료구조는 필수 메소드가 어떻게 구현되어 있는지, 런타임은 어떻게 동작하는지 알아야 합니다. (list의 필수 메소드는 set, get, pushAtEnd, popAtEnd, insertByIndex, removeByIndex, set의 필수 메소드는 insert, remove, contains? 입니다.) 자료구조 구현을 어떻게 사용하는지 알아야 합니다. 예를 들면 getNearestElementTo(x) 메소드를 구현할 수 있어야 합니다. 이 메소드 즉, x와 가장 가까운 값을 찾는 구현을 하려면 이진 검색트리를 알아야 합니다.

이 문제를 해결하는데 이런 내용을 알아야 합니다.

* 이진 검색트리 구현에 균형을 맞추는 코드가 필요하다는 점을 알아야 하지만 세부 내용은 몰라도 괜찮습니다. (선택 자료: 자기 균형 BST을 어떻게 구현하는지 빠르게 배우고 싶다면 [이 트립](http://shlegeris.com/2016/07/02/hash-ordered-treaps)을 참조하세요. 어떻게 레드블랙 트리가 동작하는지 이해하고 싶다면 [좌편향 레드블랙 트리](https://www.cs.princeton.edu/~rs/talks/LLRB/LLRB.pdf) 또는 2-3-4 트리를 배우세요.)
* 큐를 [스택 두 개로 구현](https://stackoverflow.com/questions/69192/how-to-implement-a-queue-using-two-stacks/69436#69436)할 수 있다는 점을 알아야 합니다.

다음 알고리즘은 어떻게 구현하는지 알아야 합니다.

* 그래프 알고리즘: 너비 우선 탐색(breadth first search), 깊이 우선 탐색(depth first search), 다익스트라 알고리즘 (dikstra’s algorithm)
* 빠른 정렬 알고리즘 하나. 병합 정렬(mergesort) 또는 퀵 정렬(quicksort)
* 배열에서 수행하는 이진 검색. 이 알고리즘은 제대로 작성하기 매우 까다롭고 대략적으로 알고리즘을 이해하고 있더라도 코드로 작성해볼 가치가 있습니다.

그리고 Big O 표기법도 대충이라도 편하게 사용할 수 있어야 합니다.

이 모든 내용을 어떻게 배워야 하나요? 제가 가장 좋아하는 자료는 Skiena의 Algorithm Design Manual입니다. 위에서 언급한 모든 내용을 챕터 2~6에서 다룹니다. 이 책을 좋아하는 이유는 저술 방식이 참여를 유도하고 각 부분에서 중요한 자료에 잘 초점을 맞추고 다루는데 이런 방식은 중요하다고 생각합니다. 이 책은 인터넷에서 [무료](https://github.com/addyrookie/Depot-App/raw/master/gmail/The%20Algorithm%20Design%20Manual%202ed%20%20by%20Steven%20S.%20Skiena.pdf)로 찾을 수 있습니다. 이 책의 단점은 예제가 C로 작성되었다는 점인데 C를 읽지 못하는 개발자라면 접근성이 좋지 않습니다. 저는 챕터 1~6, 12는 꼭 읽어야 한다고 생각합니다. 이 부분은 인터뷰에서 나올 가능성이 극히 낮지만 필요 없다고 생각하는 부분이 진정 핵심적인 부분을 잘 보강한다고 생각하기 때문입니다.

이런 부분에 대략적인 설명을 보고 싶다면 [Craking the Coding Interview](https://www.amazon.com/Cracking-Coding-Interview-Programming-Questions/dp/0984782850/ref=sr_1_4?s=books&ie=UTF8&qid=1530661245&sr=1-4&keywords=cracking+the+coding+interview)와 [InterviewCake.com](https://www.interviewcake.com/)의 설명이 좋습니다.

저는 Skiena의 책이 극단적일 정도로 건조하고 딱딱한 유명 CLRS 교재보다 낫다고 생각합니다.

[그래프 알고리즘에 대한 글](http://shlegeris.com/2016/07/02/graph.html)을 쓴 적이 있는데 참고가 되었으면 좋겠습니다.

**표준 알고리즘 기술**

여기까지 인터뷰에 핵심적으로 필요한 부분을 확인했습니다. 이제 다른 종류의 프로그래밍 기술로 무엇을 테스트하는지 확인하고 제가 선호하는 학습 자료도 함께 확인합니다.

이런 기술에 있어서는 Cracking the Coding Interview(이하 CtCI) 책이 가장 유용합니다. 이 책에 대해서 [작성한 글](http://shlegeris.com/2016/06/22/ctci)입니다.

알고리즘 면접 문제 중 가장 일반적이고 중점적으로 다뤄지는 요소는 다음과 같습니다.

* 동적 프로그래밍: Skiena 책의 챕터 8 또는 CtCI에서 이 주제의 챕터에서 학습합니다.
* 재귀: CtCI에 이 주제에 대한 멋진 챕터가 있습니다.
* 유명 자료 구조를 반복(iterating)하는 문제: CtCI에서 각각의 자료 구조를 다룰 때 이 문제도 함께 다룹니다. 예를 들어 BST에서는 CtCI 트리 챕터를 참고할 수 있습니다.
* 문제 해결을 위해 빠른 자료 구조를 조합하기: 이런 문제에 대한 예제는 [이 글에서](https://github.com/bshlgrs/data-structure-composer/)확인할 수 있습니다.

CtCI에서 살펴볼 수 있는 많은 문제를 살펴보는 방법이 제가 드리는 가장 주요한 조언입니다. 이 문제에서 가장 중요하다고 생각하는 부분은 위 목록과 같습니다.

이런 부류의 문제를 어떻게 학습하는지에 대해 일반적인 생각은 이렇습니다. 제 생각엔 답안을 “훔쳐보는” 일은 그래도 괜찮다고 생각합니다. 면접 문제 푸는 일을 내던지고 아예 포기하는 것보다는 문제 풀다가 막히면 해결책을 보는 방법이 차라리 나은 접근이기 때문입니다.

**알고리즘 면접에서 성공하기 위한 비기술적 측면**

이런 질문은 실제로 부담되는 환경에서 답하는 연습을 해야 합니다. 진짜 사람이 질문하는 상황에서 말이죠. 이 부분에 대해서는 [Triplebyte의 블로그 포스트](http://blog.triplebyte.com/how-to-pass-a-programming-interview)에서 다루고 있고 2, 3, 7번을 읽어보기 바랍니다.

**알고리즘과 자료구조에 대해 더 배우기**

취업 목적 학습을 넘어서 본인을 위해 즐겁게 알고리즘과 자료구조를 배우고 싶다고 가정해봅시다. 어떻게 더 배워야 할까요?

가장 쉬운 방법은 위에서 필수로 배워야 한다고 한 핵심 자료 구조에 포함되지 않는 자료 구조 중 상대적으로 간단한 자료 구조를 학습하는 방법입니다. [트립](http://shlegeris.com/2016/07/02/hash-ordered-treaps), [스킵 리스트](https://en.wikipedia.org/wiki/Skip_list), [증강 이진검색트리](http://www.bowdoin.edu/~ltoma/teaching/cs231/fall09/Lectures/10-augmentedTrees/augtrees.pdf), [서로소 집합 자료구조](https://en.wikipedia.org/wiki/Disjoint-set_data_structure)가 그 예로 모두 쉽게 이해할 수 있는 편이며 모두 멋진 알고리즘입니다.

자료구조 주제 중 이해하기 어렵지만 노력해서 이해하면 좋은 주제도 있습니다. 예를 들어 [이 슬라이드](http://web.stanford.edu/class/cs166/lectures/05/Small05.pdf)에서는 이진트리와 2-3-4 트리를 설명합니다.

흥미로운 자료 구조를 배울 수 있는, 제가 좋아하는 자료는 다음과 같습니다.

* Skiena의 챕터 12와 이후 챕터
* 스탠포드의 멋진 강의인 [CS166](http://web.stanford.edu/class/cs166/). 이 강의의 슬라이드는 멋지고 읽기 좋은 편입니다. 저는 여기서 다룬 문제가 즐거웠습니다. 자료구조와 더 놀고 싶다면 [이 프로젝트 아이디어 핸드아웃](http://web.stanford.edu/class/cs166/handouts/090%20Final%20Project%20Requirements.pdf)을 추천합니다.
* 저는 [이런 작업](http://shlegeris.com/2016/06/16/kth-richest)처럼 그다지 어렵지 않은 자료 구조 문제를 아마추어 활동으로 재미삼아 한다는 점이 자랑스럽습니다. 이 문제를 풀기 위한 해결책으로 고급 자료구조에서 얻은 몇 아이디어를 적용했다는 점이 멋지지 않나 생각합니다.