Algorithm Study Epilogue

Yonghyun Hwang, Heesoo Kim, Jean Suh, Kwanho Ryu

July 2018

Agenda

- Algorithm Study Motivation
- What to Study
- Algorithm Study
- Interview Tips
- Interview Preparation Trend
- Conclusion
- Epilogue of Study Members

Motivation for Algorithm Study

- Drill Interview Challenges
  - Many failures without fruitful feedbacks
  - Lots of stress due to unpredictable process
- Want “Deterministic” Offer
- Hope more Korean Engineers
- Hard to Get Self-Motivated
- No Known Study Group for Interview Preparation

What to Study

- Three Categories for Study
  - Computer science common
    - Coding (with c/c++, java, python, …)
    - Algorithm & data structure
  - Computer science core
    - Computer architecture, OS, compiler, ...
  - Domain knowledge
    - Big data, ML, firmware, file system, ...
- Algorithm Study
  - Core: knowledge based
  - Domain: for senior engineer
  - Targeting CS common only

Study Plan

- Original Plan
  a. Algorithm study (30%)
     - Solid fundamental before jumping into interview questions
  b. Solve interview questions along with mock interview (40%)
     - Leaked + popular interview questions
  c. Real projects (30% + hopefully more)
     - Coding & Design focus by doing real hands-on
- Actual Study
  a. Algorithm study (60%)
  b. Solve interview questions (40% + more)
  c. Real project (0%)

Algorithm Study

Study MIT Opencourseware

- MIT Opencourseware
  - Introduction to Algorithms
    - Insertion/merge/heap/counting/radix sort
    - Binary search tree, AVL tree, AVL sort
    - Hashing, table doubling. Open addressing
    - BFS/DFS. Topological sort. Dijkstra, bellman-ford
    - Dynamic programming
  - Design and Analysis of Algorithms
    - Divide & conquer, Advanced DP
    - Greedy algorithm, Augmentation: range trees
    - Cache-oblivious algorithms

Algorithm Study

Study Format

- MIT Opencourseware
  - Introduction to Algorithms: +20 lectures. Each 50 mins long
  - Design and Analysis of Algorithms: +10 lectures. Each 80 mins long
- Meet Every Week
  - 2 hours. Around 8 months
  - Before study: watched 1 or 2 lectures
  - During study: one summarized the lecture. The others asked questions if any
    - Every study members took a turn for summary
  - After study: practiced interview questions from time to time
- Review
  - After completing MIT courses, did quick review

Algorithm Study

Crunch Mode for Interview Questions

- Interview Question Practice
  - Met __every__ night. No holidays. Up to 2 hours
  - First few weeks
    - Leetcode (dynamic programming question, hard level. 4 questions)
    - Careercup (4 questions), Topcoder (1 question)
  - Later
    - Leetcode hard and medium
  - __not__ meet to solve/discuss the questions.
  - More focused on key idea and then discuss pseudo-code
  - Round table and/or anyone with idea
    - Used white board to discuss

Interview Tips

Few Focus Areas on Data Structure

- Review Basic Data Structures
  - Array, list, tree, queue, stacks, and hash (+ graph)
  - Some interview questions require two or more data structures.
- Master hash data structure (many tricky questions use hash)
  - Find an item for a given key
  - Group items by a given key
  - Distribute items to a server using hash value
  - Quick data comparison
    - Git
    - Multimedia data

Interview Tips
WhiteBoard Coding

- Why it’s challenging?
  - No syntax highlighting & checking. No __undo__. No code completion
  - Slow to write. High cost to re-write. Not accustomed to hand writing. (burn brain power)
- Key is to minimize mistake + code writing. How?
  - Before coding, discuss enough to draw function requirement (even hidden)
  - In case of anything unclear, ask to clarify everything before coding
  - Come up with pseudo code on one side first. Then, validate approach with interviewer
    - Try to come up with pseudo code with less if/else conditions
    - Use as many helper functions as possible if logic looks complex
  - List up possible corner cases along with the pseudo code

Interview Tips
WhiteBoard Coding (cont')

- Clarify Function Prototype First
  - Assume you document a function for api users.
  - Agree on Pre/post conditions for the function along with requirement analysis
  - In & out value's constraints
- Define as many functions as possible
  - Almost pseudo-code level
  - Maybe some functions doesn't need to be implemented. :)
- Too many if/else or Too many lines of code
  - maybe not a good sign in general
- In case of bad coding convention (to reduce writing)
  - Please talk first to let interviewer know interviewee is aware of that

Coding Example *(Singly linked list)*

```
// assume that entry is always found
ListNode * remove_list_entry(ListNode * head, ListNode * entry)
{
    ListNode * prev = NULL;
    ListNode * walk = head;

    // walk the list
    while (walk != entry) {
        prev = walk;
        walk = walk->next;
    }

    if (!prev)
        head = entry->next;
    else
        prev->next = entry->next;

    return head;
}
```

Coding Example

Less If/Else: Optimized to Avoid If/Else

```
// good taste code example
// fewer conditions we test for, the better code taste is
ListNode * remove_list_entry2(ListNode * head, ListNode * entry)
{
    ListNode ** indirect = &head;

    // walk the list
    while ((*indirect) != entry) {
        indirect = &(*indirect)->next;
    }
    // delete the node
    *indirect = entry->next;

    return head;
}
```

Coding Example

No If/Else: out-of-box thinking

```
// tricky code capturing essence of the problem
void remove_list_entry3(ListNode * entry)
{
    // assume: entry->next != NULL
    entry->item = entry->next->item;
    entry->next = entry->next->next;
}
```

Interview Tips

Dynamic Programming (DP) & Recursion

- DP: Master recursion first. Then, go to DP
  - Many algorithm question requires DP solution
  - First step to master DP is to master recursion
- Recursion Practice
  - Convert existing simple iterative algorithms to recursive algorithms
  - First assume that we already have a solution for n-1 inputs.
  - Then, think about how we can extend the solution for n-1 to n inputs
    - e.g., sum 1 to n. If we know sum up to n - 1, then sum up to n would be easy
- DP Practice
  - Come up with recursion first
  - Memoize anything repeated (usually memoize n, n-1, n-2, …)
  - Optimize solution if required

DP Example

Fibonacci: Recursion

```
// Compute Fibonacii sequence
// F(n) = F(n-1) + F(n-2) where F(0) = 0, F(1) = 1
int64_t fibonacci(int64_t n)
{
    if (n <= 1)
        return n;
    else
        return fibonacci(n - 1) + fibonacci(n - 2);
}
```

DP Example

Fibonacci: DP

```
#define MAX_FIB_SEQ 100
int64_t fib_array[MAX_FIB_SEQ] = {0, 1, [2 ... MAX_FIB_SEQ - 1] = -1};
int64_t fibonacci_dp(int64_t n)
{
    if (fib_array[n] >= 0)
        return fib_array[n];
    else {
        fib_array[n] = fibonacci(n - 1) + fibonacci(n - 2);
        return fib_array[n];
    }
}
```

DP Example

Fibonacci: DP Optimized

```
int64_t fibonacci_dp2(int64_t n)
{
    if (n <= 1) return n;

    int64_t n_1 = 0, n_2 = 1;
    int64_t fib = n_1 + n_2;
    for (int64_t i = 0; i < n - 2; i++) {
        fib = ((n_1 = n_2) + (n_2 = fib));
    }
    return fib;
}
```

Interview Tips

Recommendations for Algorithm Study

- Recommendations
  - Define period & Practice intensively
    - One or two questions per a day are for good maintenance.
    - But, good news is.. Can observe progress in a month.
  - Discuss with others is key to success
  - Try to solve as many questions as possible. “For yourself” seems to be a must
  - Don’t get bothered to repeat the same problems
    - Just once/twice, usually forget..
    - Important to understand train of thought
  - If some questions cannot be solved, keep them in your note and categorize them
    - E.g., hash group-by along with array to hash to random access

Interview Preparation Materials

- Google drive
  - study materials for Algorithm Study
- Books
  - elements of programming interviews in python, cracking the code interview
- https://www.educative.io/collection/5668639101419520/5649050225344512
  - Grokking the System Design Interview
  - Just like "수학의 정석"
- https://leetcode.com/
- https://www.careercup.com
- https://www.hackerrank.com/

Services for Interview Preparation

Interview kickstarter

- Homepage: http://www.interviewkickstart.com
- Cost is $4500. Around 2 months.
- Two lectures per a week. (by x googler, facebook, …)
  - First week is for orientation (why we are doing this)
  - Similar to lectures in university. Interactive
- Mainly focused on Popular topics for job interview
  - 50% algorithm and data structure
  - 50% system design
- One topic per each weeks. (e.g., array, list, hash, …)
  - Concept first, then its application in real interview questions

Services for Interview Preparation

Interview kickstarter (cont')

- On every lectures, coding practice. Homework after lecture.
  - After coding, discussing solutions. TA session for homework
- 1 on 1 meetings for progress check. (challenges, any issues, concerns…)
- Workshops on networking, Linkedin, and resume writing
  - Lecturer is from HR background
- 12 mock interviews
  - Run by x googler, facebook, linkedin, …
  - Can practice phone interview
  - Exactly same as real interview
  - After the mocks, one link is sent, which has
    - Video record on the mock
    - Feedback

Services for Interview Preparation

Only in Chinese

- https://www.bittiger.io
  - Very similar to interview kickstarter
  - Online and offline classes (so, select if interested)
  - Covers UX/UI
- https://www.mitbbs.com
  - Community based. Special community for interview preparation
- http://www.1point3acres.com/bbs
  - Interview oriented (annual salary info. __hot__ interview questions)
- Private tutoring
  - Very expensive (around 20k), but..
  - Less than three students per a tutor.
  - tutoring till a student gets an offer.

Services for Interview Preparation
etc

- http://courses.csail.mit.edu/iap/interview/index.php
  - Hacking a google interview class material by **MIT**
- https://www.outco.io
  - Located in san francisco. Very similar to interview kickstarter (located in south bay)
- https://interviewcamp.io
  - Cheap. Online based. 4 weeks. 8 hours office hour
- http://www.interviewcake.com
  - Very high quality interview questions
- http://www.gainlo.co/#!
  - Mock interview site. Middle quality. No dedication
- https://www.pramp.com/#/
  - Peer to peer mock interview

Conclusion

- Preparing job interview is challenging
- Organize study group and attack together
  - Key person who leads (dedicated to) a study group is important
  - Participants' contribution is also a key to success for study
  - Intensive study during short period is another key
- Good news is
  - Planning Algorithm Study Season 2
    - By two members from Season 1
  - Tentative plan: complete two MIT courses before end of this year
  - Near the end of this year, hopefully coming December, start crunch.

HS - Study Group

What went well

- Basic, Basic, Basic
  - 강의 반복 청취
- 발표 준비를 통한 개념 정리
- 다른 사람들을 통한 개념 이해

What didn’t go well

- System design 준비
  - paper reading

HS - Experience

- 문제풀이 방법
  - Iteration : 동일 문제의 반복 풀이를 통한 solution 개발
  - Basic concept 위주의 solution 생각
  - Coding 속도 향상

- Interview 이후
  - 회사 선택 이전, recruiter와 연락을 통한 팀 interview 주선 요청
    - project/team culture등 파악

Jean Suh

I do have background on graph algorithm and data structure.

Was a TA for a class that heavily uses graph algorithm.

5 years of career as hardware engineer working on chip physical design.

Almost no coding experience during last 5 years.

Got a job at Synopsys as senior staff R&D.

What helped me and what not...

Information, information, information.

Keep you sane and keep you get going.

Brushed up all the basics.

Advanced courses helped for my case.

Taste of interview experiences.

New good friends~~~~

It is a lot better if you have a referral and inside information.

Most of the information works for fresh young engineers and not much for experienced.

Lack of real coding and project experiences.

System design is hard to tackle without real experience.

For experienced, selecting coding language is important.

Terminologies

Not much chance for an interview without the right fits.

KR - Take Advantage of Study Groups

- Put yourself in the right environment
  - Likely to lose motivation if you study alone
- There’s always something to learn from other people
  - In my case, I got to know different ways to approach the same problem
- Second hand experience of (따끈따끈한) on-site interviews
  - The companies’ Interview process / question type significantly varies

KR - Know what’s important

- Go over Data structure / Algorithm basics over and over again
- Then apply your solid knowledge to the questions
    - Small tricks are “Good to know” but not what interviewers want to see

KR - Let recruiters help you. Cuz they can!

- Consider recruiter as your agent
  - Their job is to have us successfully join the company
- Sometimes... internal referral <<<<<<<< recruiter’s aid
- You have a right to appeal an unfair interview!
  - The worst thing that could happen is no and it doesn’t hurt your at all!