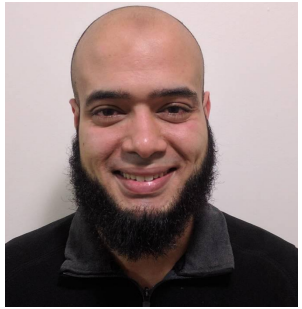
خارطة طريق
لغير خريجي حاسبات

Mostafa Saad Ibrahim
Computer Vision Researcher @ Huawei Canada
PhD - Simon Fraser University
Bachelor / Msc - FCI Cairo University

Ex-(Software Engineer / Teaching Assistant)

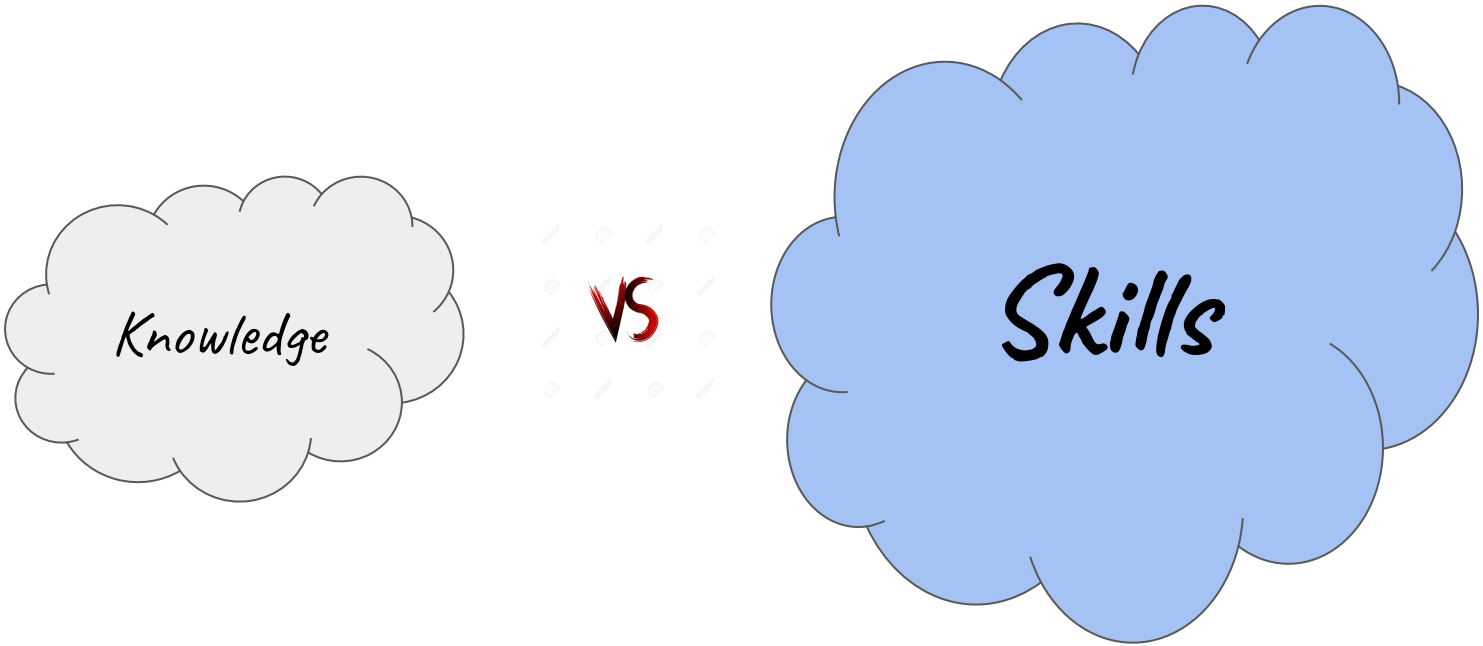


Audience

- Someone with a specialisation far from Computer Science and want to do a career shift.
- A student who is about to graduate and used to ignore studying, but now regrets and wants to find the shortest way to the market.
- Don’t try to do this plan while also busy with something else (e.g. another college)

Prerequisites

- English skills: Good or plan to be good soon
  - Reading skills: To be able to learn from books
  - Listening skills: Learning is faster from videos / online courses
  - Writing skills: For communication inside companies / applying for jobs
  - Speaking skills: Sometimes talking to English customers
  - *If your english skills is weak; don't start. Enhance English first.*
- Hard worker
  - You need 12-15 months in this guide to get in the market / 8-10 hours a day
  - Then 2 years follow up plan to cover more basics, **parallel to work**
  - You will keep learning the new fields/technologies/frameworks **forever,** like everyone


Knowledge
VS
Skills

Knowledge & Skills ladder

Img credit

Skipping fundamentals

Img credit

3 roadmaps out there!

- Full courses study, as if u were a student (3 years)
  - E.g. following a big university courses list or OSSU
- 9 / 15 months plan
  - ITI has 9 month program. Not sure of prerequisite. **Great** opportunity
  - *My roadmap is 15 months*
    - 4-6 months fundamentals: programming, oop, problem solving (algorithms/data structures)
    - 3 months: expanding some knowledge in CS courses
    - 6 months: Market technology focused
- 3-5 months diplomas
  - Some guys announce such very short courses/projects
  - I believe this is very bad choice … and most of providers targets fast turnover

Roadmap Big Picture: 3 Stages

- **Mastering** a programming language, such as Java or C#.
- **Mastering** Object Oriented Programming (OOP) Skills
- Fair understanding for **data structures**
- Some background in **Algorithms** / skills (the more, the better)

4-6 months

- Enhancing your **CS background**
  - Major courses
  - Non Major courses

3 months

- Master some **popular technology** market direction
  - Backend
  - Frontend: web or mobile [IOS, Android]

6 months

Stage 1A - The critical core

- Programming.
  - Strong in programming => Most of next things will be a piece of cake
  - You should learn several languages. During this roadmap, focus on 1 or 2
    - **Java** or C# are general purpose languages. Python also increasing in the market
  - In your first language: set a goal to read a book from A-Z (something 1000+ pages book)
  - To learn initially: Attend a course or youtube playlist
  - Programming can be thought as 2 parts: Basic syntax and OOP
- Programming:: Basic Syntax
  - Data types, Selection, Repetition, Functions (aka Methods), Arrays
  - Also: Very Basic class creation, Java Collections, Basic Files read/write
  - Now, Implement 1-2 projects of a reasonable size. Here is an example (it is from my playlist)

Stage 1B - The critical core

- Programming:: OOP
  - Objects, OOP (Inheritance and Polymorphism), Exceptions
  - Develop new 1-2 console projects using OOP (e.g. from TopCoder projects)

Stage 1C - Problem-solving [recommended]

- If you want be stronger in coding, highly recommended to do it
  - It is based on data-structures and algorithms, but learning them in incremental smooth way
- First, understand what is: Online Judge / Codeforces.
- Follow my sheet and solve levels 1 ro 3
  - You can solve level 1, then 2 and 3 later in parallel
- Stage 1C can go **parallel or before** Stage 1B
  - Going **before** actually makes you stronger

Stage 1D - Data Structures Course

- Do after Stage 1B.
  - If started after Stage 1C, you will understand **so fast**
- Study the **basic topics** (Stack, LinkedList, Queue, Binary Search Tree)
  - Write code and play with it
- Understand and use Hashtable / Sets
- The most critical outcome:
  - **When** to use a data structure?
  - What is the time **complexity** of it?
- Internet has many courses. My playlist

Stage 1E - Algorithms Course [recommended]

- If you did 1C, you won’t need to do 1E
- Learn to compute Algorithm **Order** Complexity
- Algorithms
  - Graph Theory: Representation (DFS, BFS, Dijkstra).
  - Binary Search technique
  - Trie Data structure
  - Dynamic Programming
  - Greedy
- I hate to tell you can skip this phase, but many do so and they're working

Stage 1F - More programming concepts

- You need to understand & be good in more concepts
- Complete a programming book from A-Z
  - multi-threading, client-server, crawl internet pages.
    - Better learn multithreading after some chapters in Operating systems
    - Better learn client-server after some chapters in Networking
    - Implement a client-server chat app that uses multithreading
  - Be aware of things such as Unicode, Internationalisation, Regular expressions
  - Deeper in files reading/writing/exceptions - Mastering Collections
  - Do a project or enhance the previous projects with what you learned
  - Learn basics of GUI, but don't waste time over it [optionally]
- Optional: **CS 50**

Stage 1 Summary

- **Must do 1A and 1B**
  - You must do good-size projects to build an important skill (building projects skill)
- Must understand 1D to some extent
- Optionally but recommended either 1C or 1E
  - You can also make it parallel to stage 2 and/or 3
- Proceed with 1F, but better parallel to stage 2

Stage 2: Enhancing your CS background

- Stage 2A: Core practical CS subjects (initially 3-5 chapters)
  - Networking, Operating Systems, **Databases, Software Engineering**
    - Last 2 are much more critical to go deeper, especially Databases for backend guys
    - SWE: Learn to build class and Sequence diagrams. The waterfall model, the agile.
      - Later, learn design patterns
  - Do projects/apps that utilize what you learned
  - Also Discrete Mathematics is very useful to mind & skill (enhances also 1C)
- Stage 2B: Relevant courses
  - Read initially 1-3 chapters
  - Computer Theory, Assembly programming, Computer Architecture, Computer Graphics, Cryptography, Security, Cloud Computing, Parallel Computing
  - Avoid Math (Probability, Calculus, Linear Algebra)
  - You can stage parallelize 2B with stage 3

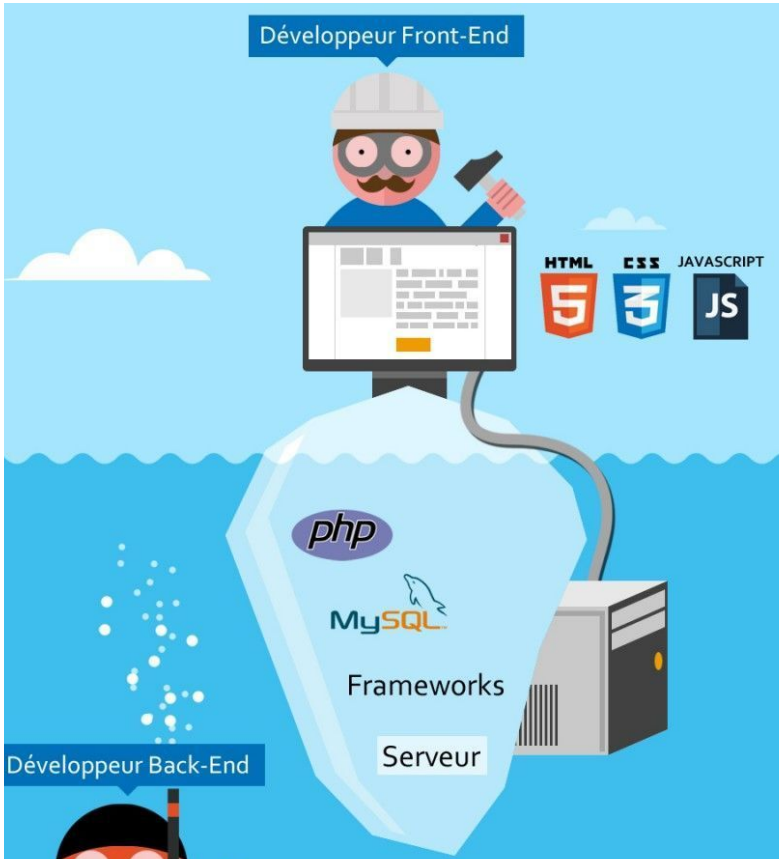
Stage 2: Materials

- Most of choices are ok. E.g. Use OSSU
- For databases: Arabic Playlist - Course - Course - Book
- Discrete Mathematics: Arabic - book

Stage 3: Market Track

- Front/Back end software engineers are the **most popular**

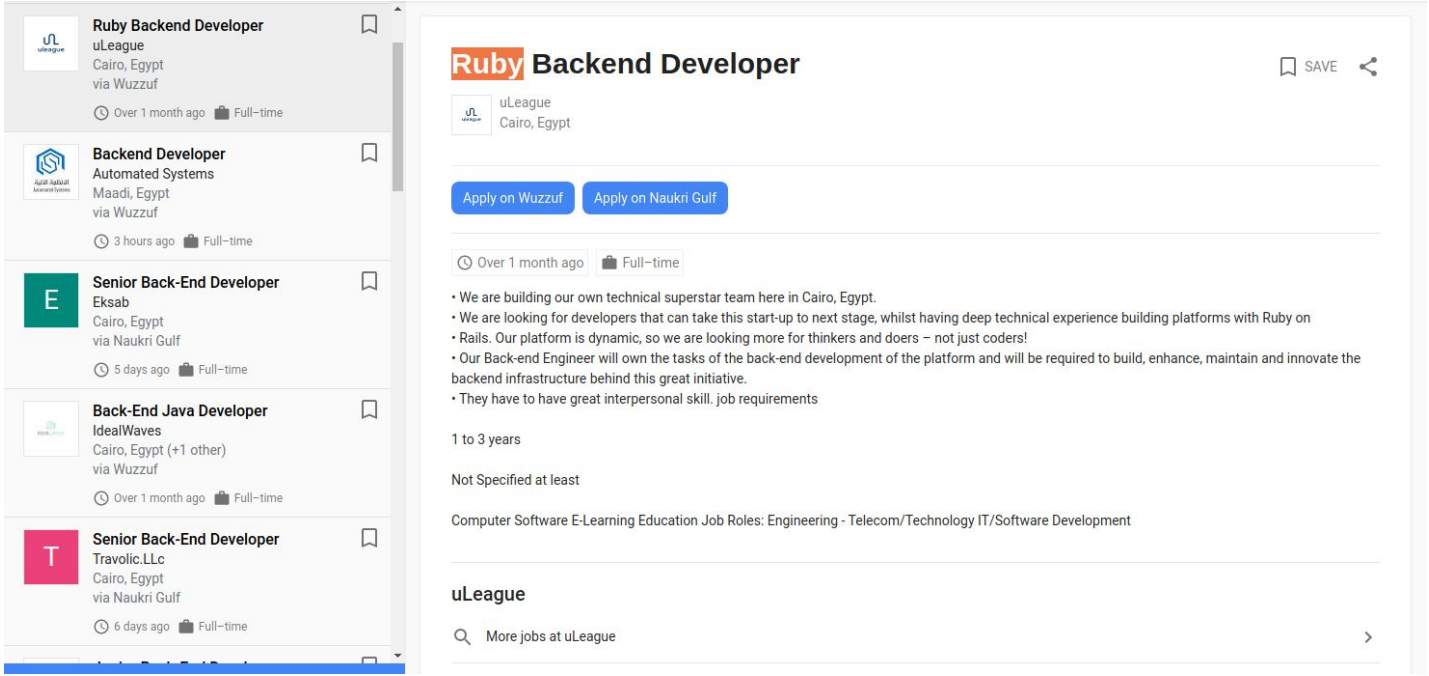
- Many nice jobs are still **limited** in Egypt (2020)
  - Data science, Machine Learning, Embedded development, Games Development, Security



Img credit

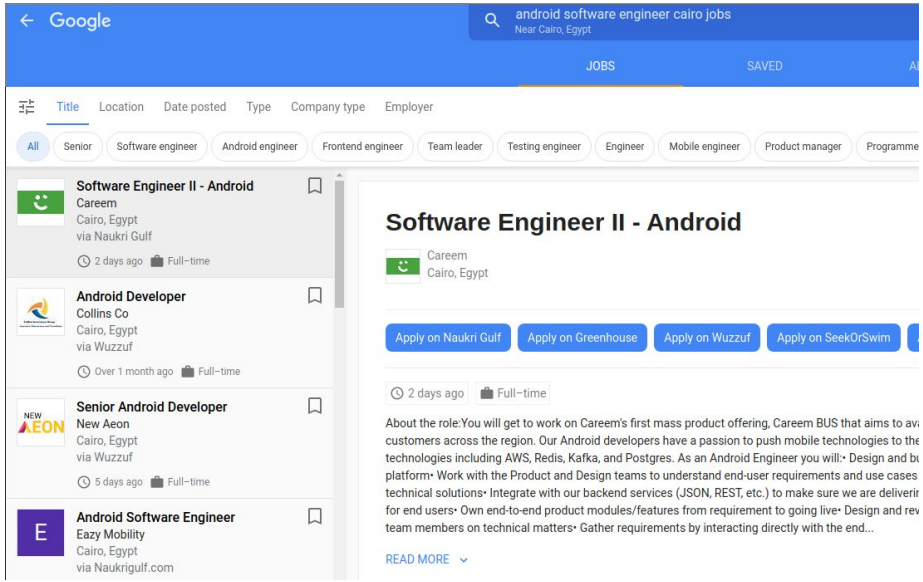
Stage 3: Search for jobs: Get sense of Technologies

- Google it, e.g. *backend developer cairo jobs*



Stage 3: Search for jobs: Get sense of Frequency

- Google it, e.g. *android software engineer cairo jobs*



Stage 3: Search for jobs: Get sense of Frequency

- Notice: Many of the opportunities are hidden
  - Handled through recommendations insider the company
  - Networking with people (in the company, college, meetups, conferences) is critical
- Use linkedin
  - See for job postings
  - Create profile, add people
    - In their connections, see what are their jobs / job history

Stage 3: Market Track - Front End

- Frontend (Web, IOS, Android)
  - Android: udacity-nanodegrees - Skill Set
  - IOS: IOS - Course - Arabic Course
  - Web: Web / Web / CS50 / Video
  - Cross Platform (e.g. Flutter, React Native)

Stage 3: Market Track - Backend

- Backend: **Core**(OOP, database, SWE design) + Technology Track
  - Database: Learn SQL very well, then later NoSql
- Several Technology tracks - Search & decide as varies over time
  - Guide
  - Php & Laravel
  - Ruby on Rails
  - Python, Flask & Django
  - Java EE (web services, JMS, JPA, ...)
  - Node.js
  - Course

Stage 3: Market Track - Full Stack

- Full Stack = Can handle both front and backend
  - Maybe better select first one area to be good at first
- Full Stack

Stage 3: Enhancing Skills

- Don’t rush in the core part - moving from technology stack to another easy
- Whatever track: Build 3 projects of big size. This is your **CV** Core
  - Apply SWE concepts as much as you could
  - Well documentation for your project
  - Upload on github with screenshots and/or demos
- Then, build linkedin account and find job

Find a job

- Send CVs to companies. Attend job fairs. Your first job will be hard to find
- In parallel, start to do freelancing: such as upwork or others
  - It takes time to understand the process
  - It takes time to get your first customer
  - You can be 15-20 dollar per hour
- Salaries
  - Some companies make use of guys without certificates or weak graduates
  - Might be paid 2000-3000
  - Proper salaries are 5000-6000 … some companies a bit more
  - Anyway. Get your ‘good’ job. Good here in terms what you will learn not the money
  - Later move to the right opportunity

Stage 4

- Now you got a job. Work in parallel to enhance your background
- Learn several other programming languages (e.g. Python, Go, Javascript)
- For every CS subject, go deeper. Do a project
- Good in several OS: E.g. Windows and Linux (ubuntu)
- Soft skills: Presentation / Communication / Writing skills ...

Challenges

- Self-discipline
- Time availability: being a student in another college
- No provided assignments/projects
- No TA to help/mark assignments/projects
- Weak connections
- No internships
- Difficulties in the first job

Concerns

- Certificate
  - No / Online / Some institute
  - Working abroad
- Age
- Tools
  - Office tools (microsoft or google ones): Word, Power Point, Excel
  - Git / Github
  - Command lines
  - Management tools: Trello.com

Notes

- You can’t learn 1-2 courses and find a job or freelance!
  - A lot of effort must be put
- You can be bad in math and still be very good software engineer!
- Learning programming in the first 2-3 weeks might be hard! But after that, normal.
- It is better to attend online courses for Programming and OOP
  - You may also attend for data structures and algorithms
  - You don’t need to attend in other

Got lost?

- The first 2 targets are fundamental and when you are good, you will understand remaining Easily (**Programming language + OOP**)
- 1- Learn programming language such as Java or C#
  - Learn from online/onsite course
  - But, Read it later from a book, page by page.
- 2- Do 2 console projects (See example here)
- 3- Learn OOP
- 4- Do 2 OOP projects (better new ones)
- 5- Learn github tool. Upload your projects with screenshots
- Now, after that read the roadmap again

Anticipating the future

- One property in this field is its dynamicity
  - Today languages/technologies will die at some time
  - Search about Javascript/Machine Learning in market size
- Few things to do
  - Keep your eyes on the new things - try them out
  - Try to understand the trends.
    - Don’t wait till something be so clear as part of the next 5 years.
  - More importantly, be a graduate with strong basics
  - Being strong in basics = Fast learner = More confidence
  - Don’t fully depends on what faculty gives. Do your homework. Go beyond that
- Skills & Responsibilities for my next career steps

Links

- 7asebat playlist
- My ask.fm
- OSSU
- ITI

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً