Yannick Müller muelleya@student.ethz.ch
yajm.ch

ETH MADE EASY

Computer Science Block 5

January 2021

Version 0.1
Letzte Bearbeitung: 28. September 2020

Diese Zusammenfassung besteht aus den Vorlesungs- und Übungsstundennotizen. Es wurde versucht den Aufbau möglichst nah an die Skripts und Vorlesungsfolien anzulehnen. In diesem Skript wurde versucht Vieles zu vereinfachen, weshalb zum Teil Sachen weggelassen wurden.
Für die Richtigkeit dieses Skriptes kann ich nicht garantieren. Viel Spass beim Lernen.

Table 1: Prüfungsplan

Datum	Zeit	Fach	Hilfsmittel	Raum
Di 06.08	14:00-16:00	Computer Networks		
Mo 10.08	15:00-18:00	Formal Methods & Functional Programming		
Sa 15.08	14:00-17:30	Introduction to Machine Learning	2 A4 Seiten LaTeX	
Di 18.08	09:30-11:30	Data Modelling & Databases		
Do 20.08	09:00-11:00	Wahrscheinlichkeit & Statistik	10 A4 Seiten LaTeX	
Sa 22.08	13:30-15:30	Rigorous Software Engineering		
Do 27.08	11:00-12:30	Discovering Management		

Ziele

1. Lernplan einhalten
2. In Gruppen lernen
3. Prüfung bestehen

"The best way to predict your future is to create it."
—Abraham Lincoln

"Live as if you were to die tomorrow. Learn as if you were to live forever."
—Gandhi

"Progress is impossible without change, and those who cannot change their mind cannot change anything."
—George Bernard Shaw

Alle ETH Made Easy Skripts

- 1. Semester - January 2019
- 2. Semester - August 2019
- 3. Semester - January 2020
- 4. Semester - August 2020
- 5. Semester - January 2021
- 6. Semester - August 2021

Vorbereitungstests

Noten	FS 16	HS 16	FS 17	HS 17	FS 18	HS 18	FS 19	HS19
APC								
Visual Computing								
Computer Systems								
Wahrscheinlichekeit & Statistik								
Neuroinformatics								
HPCSE								
HCI								

OKRs Overall it was a good performance as I reached 7 out of my 10 Objectives.

FMFP	10. Aug	Perc. 3.25	Score ✗
40 % FS 17	14. July	15%	✗
70 % of Exercises	28. July	70%	✓
50 % FS 18	7. Aug	50%	✓

Computer Networks	6. Aug	Perc. 4.5	Score ✓
50 % FS 19	21. July	31%	✗
70 % of Exercises	1. Aug	100%	✓
60 % FS 18	4. Aug	50%	✗

Wahrscheinlichkeit & Statistik	20. Aug	Perc. 3.5	Score ✗
40 % HS 17	22. July	28%	✗
Zusammenfassung	31. July	100%	✓
50 % FS 19	3. Aug	18%	✗
70 % of Exercises	11. Aug	70%	✓

Data Modeling and Database	18. Aug	Perc. 4.5	Score ✓
70 % of Exercises	23. July	100 %	✓
50 % FS 18	27. July	46%	✗
60 % FS 17	11. Aug.	44%	✗

Machine Learning	15. Aug	Perc. 4.25	Score ✓
40 % FS 17	26. July	33%	✗
Zusammenfassung	1. Aug	100%	✓
70 % of written Exercises	12. Aug	100%	✓
50 % FS 19	13. Aug	36%	✗

Rigorous Software Engineering	22. Aug	Perc. 4	Score ✓
20 % FS 19	21. Aug	40%	✓
30 % of Exercises	22. Aug	100%	✓

So I hit the necessary 70% exactely.
I got 42 out of the 54 credits I've signed up for, so this is a 78%, which is higher than expected.
I reached 19 outz of the 33 key results, which should have been a little higher; 58% instead of the anticipated 70%.

Chinesisch	25. Aug	Perc. 4.5	Score ✓
Mündlich einen Tagesablauf beschreiben	18. July	80%	✓
150 Chinesische Zeichen kennen	24. July	40%	✗
Exam-Prep. Doku. vorlesen	1. Aug	60%	✓
Mündlich sich mit jemandem verabreden	25. Aug	80 %	✓

Discovering Management	27. Aug	Perc. 4	Score ✓
Zusammenfassung durchlesen	25. Aug	60%	✓
Slides durchgegangen	26. Aug	100%	✓

Ironman 70.3 under 5 hours	13. Sep	Perc. 4:55	Score ✓
250 Watt Cycling Average for 1 hour	15. Aug	100%	✓
Between 7.5% - 8.5% Body Fat	22. Aug	8.6 %	✗
Race Nutrition Plan + Equipment to Store	29. Aug	100%	✓
4 Swim-Bike Transitions under 4 minutes	29. Aug	0%	✗
Course Visit	6. Sep	100%	✓

Personal	29. Aug	Perc. 50%	Score ✗
Finish 3 Books	23. Aug	33%	✗
Ferien Planen September	26. Aug	100%	✓
Computer Cleanup + Sway Setup	29. Aug	50%	✗

	Monday	Tuesday	Wedensday	Thursday	Friday	Saturday	Sunday
6:00 - 7:00		Swimming					
7:00 - 8:15							
8:15 - 9:00	Chinesisch LEE D 105	HPC Homework CSE-Lab	Chinesisch LEE D 105	V Neuroinformatics Zoom	E HPC ML H44 Zoom 847282	V HPC Rewatch Polybox	
9:00 - 9:15							
9:15 - 10:00	Chinesisch	HPC Homework	Chinesisch	V Neuroinformatics	E HPC	V HPC Rewatch	
10:00 - 10:15							
10:15 - 11:00	V Comp Systems ML D 28	V Visual Computing Teams	V APC Rewatch Moodle	E Neuroinformatics Zoom	V Comp Systems ML D 28	Entrepeneurship	
11:00 - 11:10							
11:10 - 11:55	V Comp Systems	V Visual Computing	V APC Rewatch	Cycling	V Comp Systems	Matteo Nachhilfe	
11:55 - 13:15							
13:15 - 14:00	Tandem Chinese	E Visual Computing Teams	Vocabulary Chinese	Homework Chinese OLAT	Tandem Chinese	APC Homework TI	
14:00 - 14:15							
14:15 - 15:00	V APC Zoom	E Visual Computing	E APC Zoom	V Visual Computing Teams	E Computer Systems NO D 11	APC Homework	
15:00 - 15:15							
15:15- 16:00	V APC	E Visual Computing	E APC	V Visual Computing	E Computer Systems	WuS	
16:00 - 16:15							
16:15 - 17:00	G Entrepreneur HG F 3	Cycling	Fitness	NeuroInf Homework OLAT	Fitness	WuS	
17:00 - 17:15							
17:15 - 18:00	G Entrepreneur	Chinese Homework OLAT	CS Homework Moodle	NeuroInf Homework	CS Homework Moodle	Vocabulary Chinese	
18:00 - 19:00							
19:00 - 20:00							
20:00 - 21:00							

ACP - Lecture

Prof. Brnd Gärtner

Contents

The following was presented in the lecture on 15. September 2020.

Definition 1. Bootstraping is to solve a problem out of nothing. So recursivly apply the method you are about to develop.

0.1 MST in Expected Linear Time

Algorithmus 0.1: Kruskla

Sort Edges by increasing weight; add edges in this order, skipping edges that close cycles. $\mathcal{O}(mlog(m))$

Algorithmus 0.2: Boruvka

Based on the fact that for each vertex the incident edge of smallest weight is in the MSF. For each vertex compute the incident edge of smallest weight, and add them to the MSF. Contract all these edges and recurse. $\mathcal{O}(mlog(n))$

Definition 2. Let T be a spanning forest. Adding an edge e would close a cycle. e is T-heavy if e is the edge of largest weight in this cycle.

Lemma 0.1

Given T, a spanning forest of G, the T-heavy edges can be found in time $\mathcal{O}(n+m)$

Algorithmus 0.3: Improved MSF algorithm

1. Perform 3 iterations of Boruka $\Rightarrow$ New Graph $|v| \leq \frac{n}{8}$
2. Select each edge of G with probability $\frac{1}{2} \Rightarrow G'' = (V', E'')$
3. Call improved MSF algorithm $\Rightarrow$ MSF $T = (V', E‘‘‘)$
4. Compute the T-heavy edges $HsubseteqE'\mathbb{E}‘‘‘, L = E'\backslash E‘‘‘\backslash H$ the non T-heavy edges
5. Form $G‘‘‘ = (V‘, E‘‘‘ \cup L)$ and recursively call improved MSF algorithm on $G‘‘‘ \Rightarrow MSF(G‘) \Rightarrow MSF(G)$

The following was presented in the lecture on 21. September 2020.

1 Minimum Cuts

Definition 3. A cut with the smallest possible number of edges is a minimum cut. Can find it in $\mathcal{O}(n^2)$ time.

Definition 4. Mutligraph is a graph where two vertices may be connected by more than one edge.

Algorithmus 1.1: Basic Min Cut

While G has more than 2 vertices do:

1. Pick a random edge e of G
2. G := G/e

return the size of the minimum cut in G (number of edges connecting the two vertices)

Lemma 1.1

G is a multigraph with n vertices and a random edge e, then:

$$prob(\mu(G) = \mu(G/e)) \geq 1 - \frac{2}{n} \tag{1}$$

So the probability of success is $p_0(n) = \frac{2}{n(n-1)}$
⇒ Algorithm almost always fails. But we can run it several times (N times) and in the end output the smallest cut that was found. This fails only if all N attempts have failed, which is at most

$$1 - \frac{2}{n(n-1)} \leq e^{-\frac{2N}{n(n-1)}} \tag{2}$$

1.1 Bootstraping

Basic MinCut is more likely to fail when G has become small ⇒ stop contracting at some point and use other algorithm.

Algorithmus 1.2

repeat N times:

1. H := RandomContract(G,t)
2. call $A_i(H)$

end return the smallest of the n cut sizes obtained
Runtime: $\mathcal{O}(n^{2+\epsilon})$

Algorithmus 1.3: MinCut(G)

if $n \leq 16$ solve
else

1. t := $\lceil \alpha n \rceil + 1$, for a constant $0 < \alpha < 1$

2. H_1 := RandomContract(G, t)

3. H_2 := RandomContract(G, t)

4. return min(MinCut(H_1), MinCut(H_2))

end

The following was presented in the lecture on 23. September 2020.

2 Search Tree

Definition 5. A search tree (for a set S of n distinct real numbers, keys); a tree with n nodes (each node contains a key) such that for every node the keys in the left subtree are smaller and the ones in the right subtree are larger.

2.1 Random Search Tree

Definition 6. In random search trees the x element is choosen uniformly at random. Each element is equaly likely to become the root of the subtree. So this gives a certain probabilty that a certain tree arises.

Note 1. Random choices in subtrees are independent from each other and from the root.
The distribution is the same as the one we get when we insert the elements of S in random order (random permutation each one chosen with prob $\frac{1}{n!}$) into an inityally empty tree.

2.1.1 Expected Depth of the smallest Key

$$d_n = \mathbb{E}[D_n] = \mathbb{E}[D_n^{(1)}] \leq lnn \tag{3}$$

Theorem 2.1

The expected overall depth of a random search tree for n keys is

$$2(n+1)H_n - 4n = 2n \ln n + \mathcal{O}(n) \tag{4}$$

In particular, the average depth is on expecation $2 \ln n + \mathcal{O}(1)$

The following was presented in the lecture on 28. September 2020.

Theorem 2.2: Jensen's Inequality

If f is a convex function, then

$$f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)] \tag{5}$$

Theorem 2.3: Markov Inequality

X nonnegativ random variable with finite expectation; $\lambda > 0$ Then

$$prob(X \geq \lambda \mathbb{E}[X]) \leq \frac{1}{\lambda} \Leftrightarrow prob(X \geq t) \leq \frac{\mathbb{E}[X]}{t} \tag{6}$$

Theorem 2.4

$$prob(X_n \geq t \ln(n)) \leq n^{t-1-t \cdot \ln \frac{t}{2}} \tag{7}$$

Theorem 2.5

The expected number of comparisons in QuickSort(S) is:

$$2(n+1)H_n - 4n = 2n \ln(n) + \mathcal{O}(n) \tag{8}$$

The following was presented in the lecture on 29. September 2020.

Algorithmus 2.1: QuickSelect(k, S)

Fint the element of rank k among a set S of n elements

1. Choose $x \in S$
2. split S into $S^{<x}, \{x\}, S^{>x}$
3. $l = |S^{<x}| + 1$ // rank of x
4. if $k < l$: return $Quckselect(k, S^{<x})$
5. if $k < l$: return $Quckselect(k - l, S^{>x})$
6. if $k = l$: return x

Theorem 2.6

The expected number of comparisons in quck select to find the element of rank k among n elements is at most 4n

Definition 7. A Treap (Tree + Heap) stores items, where each item x has a key $key(x)$ and a priority prio(x). Assume: no two keys / no two priorities are equal.
Treap for a set of items Q: binary tree with nodes labeled by Q, such that

- it is a search tree with respect to keys
- it is a min-heap with respect to priorities.

Theorem 2.7

In a randomized search tree (treap with priorities chosen, v.a.r. from [0,1]), operations find, insert, delete can be done in $\mathcal{O}(\log(n))$ time per operation, where n is the current number of elements.

Computer Systems - Lecture

Prof. Timothy Roscoo

Contents

1 Operating System

An OS is an interplay between multiplexing, protection and abstraction.
Application side of view: Interpaly between a referee, illusionist and glue.

Example 1. What? CPU, Who? Processes, What does it? Scheduler
What? RAM, Who? Processes, What does it? Virtual Memory System
What? Dataset, Who?

Note 1. A program interacts with the OS via a Transfer or a Switch.

1.1 Exception

1. Process completes current instruction
2. Enters Kernel Mode
3. Jumps to Predefined Handler

After that do excactly one of the following:

1. Resume next insturction after exception (Trap)
2. Restart instruction that was executing (Fault)
3. Other

1.2 Types of exceptions

Definition 1. Synchronous: Caused by a program. e.g. Page Fault, System call

Definition 2. Asynchronous: External to the core. e.g. Interuptions

1.3 Why use Mode change in an OS?

1. Hides the OS behind an interface
2. Protects Hardware from programs

Traps Give the illusion of the hardware

Faults Hardware can be reconfigured without the programs knowledge transparenty to software.

Interupts OS can take control at any time

Referee Scheduling, Memory Allocation, Memory Protection

Glue Device Access, Programm Execution

Illusionist Provide virtual resources to user-space, virtual memory, shared network interface, paging

Note 2. On top of that, a mode switch doesn't need to return back to the user libary mode, but it can also do a context switch.

The following was presented in the lecture on 25. September 2020.

1.4 What is a process

Definition 3. A process is

1. a running program.
2. Threads of execution
3. Name Spaces
4. Kernel State
5. Resource Principal
6. Security Principal

1.5 What does a process do

It virtualizes the machine.

1.6 Execution Environment

1. Instruction Set
2. Kernel ABI System Call Interface
3. Virtual Address Space
4. Interrupts (Signals in Unix are called Upcalls)

1.7 Process Life Cycle

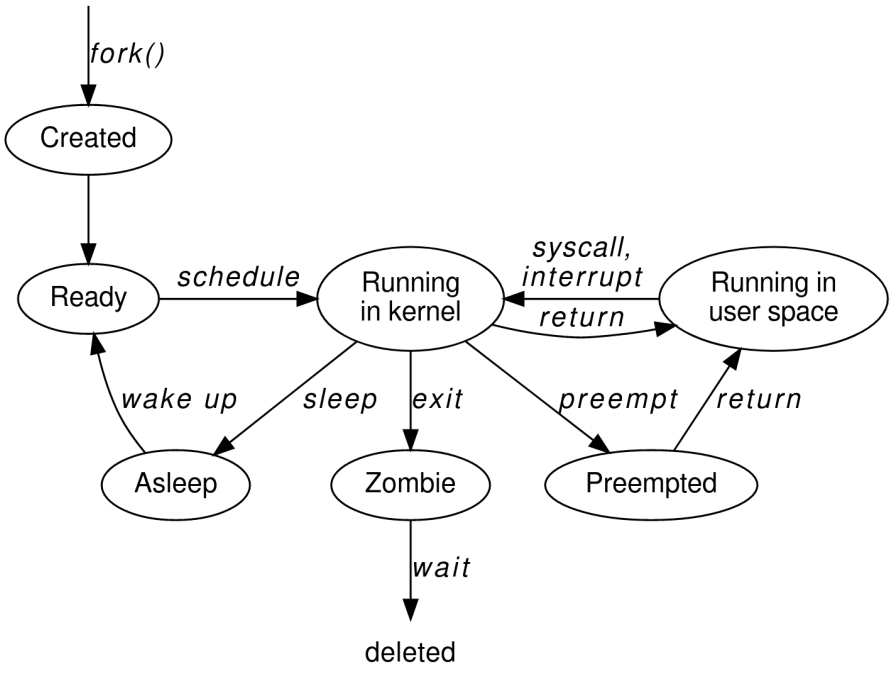

Figure 4.9: Simplified lifecycle of a Unix process

The following was presented in the exercise on 25. September 2020.

Kernel Special process that runs in privileged mode, typically event driven server

System libary Convenience function, System call wrappers

Deamon Processes whcih are part of the kernel but live in user-space, provides modularity, fault-tolerance

Definition 4. Exokernel, move functionality into system libraries instead of user-space

Definition 5. Mutlikernel: run different kernels on different cores (E.g. Barrelfish)

1.8 Bootstrapping

1. Power on
2. Load Bios from ROM to memory
3. Bios loads boot loader into memory
4. boot loader loads OS (kernel) from disk
5. Transfer control to OS

Note 3. A synonym different name for one object and a homonym is same name for different objects

Note 4. An IP address is not a pure name.

Note 5. A symbolic link points to another name to another link

Note 6. An advantage of running a process in userspace is security, memory illusion

The following was presented in the lecture on 28. September 2020.

1.9 Shared Memory (Centralized)

- Hardware Synchronization
- Spin Locks
- Transactional Memory

1.10 Messages (Decentralised / Distributed)

- Pipes (Synchronous / Asynchronous)
- Signal (Upcalls)
- Client / Server (Named Pipes / Remote Procedure Call)

Note 7. It creates a pipe first and then forks the process. So both processes know which pipe they need to use to talk to each other.

Visual Computing - Lecture

Prof. Pollefeys Marc

Contents

The following was presented in the lecture on 22. September 2020.

1 Cameras

1.1 Prism

Seperate Red, Blue and Green with two Prism and then measure the light which came through.

1.2 Filter mosaic

Uses a Bayer filter and one coats the filter directly on the sensor. But one needs aliasing to reconstructe the pixels which where not measured.

1.3 color CMOS sensor

Measure with different filter stacked on each other.

Note 1. Segmentation is the ultimate classification problem. Once sovled, Computer Vision is solved.

The following was presented in the exercise on 22. September 2020.

Definition 1. Region growing means that one starts at a seed point and then adds neighboring pixels that share some properties and then iterate with the newly added pixels.

1.4 Bluescreen

Definition 2. Mahalanobis distance

$$(x-\mu)^T \sum^{-1} (x-\mu) > t \tag{1}$$

Covariance Matrix $\sum_{ij} = E[(X_i - \mu_i)(X_j - \mu_j)]$

Estimation from n data points $\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})(x_i - \overline{x})^T$

1.5 Pixel-wise Color Model

The following was presented in the lecture on 29. September 2020.

2 Convelution and Filtering

2.1 Linear Filtering

$$I'(x,y) = \sum_{(i,j)\in N(x,y)} K(x,y;i,j)I(i,j) \tag{2}$$

3 Gaussian Kernel

Weight constributions of neighboring pixels by nearness

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{3}$$

1. Rotaionally symmetric
2. Has a single lobe
3. Still one lobe in frequency domain
4. Simple relationship to σ
5. Easy to implement efficiently

3.1 High-pass filters

$$\text{Laplacian operator: } \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad \text{High-Pass Filter: } \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{4}$$

The following was presented in the exercise on 29. September 2020. Convolutions are:

Operator $\star$ mapping image and kernel to images $I_{Out} = k \star I_{In}$

Local $I_{Out}[i,j]$ depends only on neighbors of $I_{in}[i,j]$

Further Properties Linear, Associative and Shift invariant

$$I'(x,y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} K(i,j)I(x-i,y-j) \tag{5}$$

3.2 Canny Edge Detection

Combine noise reduction and edge enhancement

Apply derivative of Gaussian filter

Non-maximum suppression Thin multi-pixel wide ridges down to single pixel width

Hysteresis Accept all edges over low threshold that are connected to edge over high threshold

3.3 Hysteresis

- if $G < T_{low}$ then it's definitely not an edge
- if $G > T_{high}$ then it's definitely a strong edge
- if $T_{low} < G < T_{high}$ then it is a weak edge if and only if it is connected to any strong edge through other weak edges

High Performance Computing - Lecture

Prof.

Contents

The following was presented in the lecture on 26. September 2020.

Definition 1. Concurrency is the existence of two or more stream of insturctions, whose execution order cannot be determinted a priori.

Definition 2. The existence of two or more stream of instructions executing simultaneously.

0.1 Different tyes of Parallelism in Hardware

Multiple Physical Cores Thread-level Parallelism

Pipelining Instruction-Level Parallelism

Vecotrization Data-Level Parallelism

0.2 POSIX Threads

```
#include <pthread.h>

pthread_t threads[NTHREADS];
pthread_create(&threads[t, NULL, do_work, (void *) t]);
pthread_join(threads[i], &status);
pthread_exit(NULL);

gcc -pthread -o myprog myprog.c
```

0.3 C++11 Threads

```
#include <thread>

std::thread threads[8];

threads[i] = std::thread(do_work, t);
threads[i].join();
```

0.4 OpenMP / Open Multi-Processing

1. Pre-compiler directives
2. Runtime libary calls
3. Environment variables

```
int main(int argc, char* argv[]) {
  int results[100];

  #pragma omp parallel for \\Parallelization in OpenMP
    for(int i=0; i<100; i++)
      do_work(results[i]);

  return 0;
}
```

Neuroinformatics - Lecture

Diverent Professor

Contents

The following was presented in the lecture on 1. Ocotber 2020.

1 Resting Potentials

1.1 Nernst equation

$$V_{eq} = \frac{k_B T}{zq} \ln\left(\frac{[X]_{out}}{[X]_i n}\right) \tag{1}$$

Startup - Lecture

Anil

Contents

1 Questions

: Team is really important. How to hire the right person for your startup?

2 Kind of Startups

2.1 Platform

Platform Startup need a lot more money. Need investors who do not look after patents. Switzerland / Europe investors are not into that. The characteristics are tech-agnostic own the vertical impacted by region. The opportunity of scaling to a huge user base makes it that the startup needs hundreds of million of dollors. Usually the first one is the winner, that needs a lot of money.
US investor are Financial investors who want to make a lot of money fast and therefore put a lot of money in it, more risky.
European investor are usally strategic invesors which can take up to 3 years.

Scaling is expensive ⇒ No interested investors in Europe

2.2 Pain

The characteristics is that there is a gap between the current technology which is not yet addressed (Cannibalise their own business)

2.3 Technology

Bring technology to market. Timeline driven by tech.
Replicability is the key success factor

2.4 Bubble

Never assume that your opposite knows the same. You're usually in a knowledge bubble. Therefore it is important to address your customer's problem in its thinking space.

People pay for the percepton of risk

2.5 Investor Advice

1. Check virality

2. Check timing
3. Check success fators
4. Be humble to learn a new profession

You don't need to know where the puck is now, know where the puck is in three years from now.

Luck and right timing

Definition 1. Team needs to have ambition, persistence and honesty.

Definition 2. The Market and business model needs to be scalable, disrupte and leverage.

Definition 3. The technology needs millions to scale up and the right product for a market fit. FOCUS!

Note 1. There are 12 to 16 jobs in a startup. Minimal of 2 person, but no work life balance, then raise 300'000 CHF. With ideal 6 person one needs to raise at least a million.

Note 2. Always to bottom up Calculation. How many sales person do you need to sell this many water bottles.

Note 3. Tell the investor some achievment and they will check if you have done nothing bad in the past. Don't screw people over, because the 900 investors in Switzerland know each other. Co-Founders should have Ethics and Trust, rather than skills.

Note 4. Investors like high returns, affinity to business and a active role.

Networking is one of the most important skills as an entrepeneur

People want to do something meaningful, so maybe one can even recrut a Senior Vice President. nicolos redalpine