

Informatica

2^e semester: les 11

Hashing & Internet Slot

Jan Lemeire

Informatica 2^e semester
februari – mei 2021



Vrije Universiteit Brussel

Vandaag

1. Hashing
2. Internet (deel II)
3. Examen

Hoofdstuk 9

Hashing

Performantie datastructuren

Datastructuur	Random access (opvragen i ^{de} element)	Find (via naam)	Toevoegen / verwijderen
Array	$O(1)$ ++	$O(n)$ $O(\log(n))$ als gesorteerd	$O(n)$ -
ArrayList	$O(1)$ ++	$O(n)$ $O(\log(n))$ als gesorteerd	$O(n)$ -
Linked list	$O(n)$ -	$O(n)$ --	$O(1)$ ++
Binaire boom	n.v.t.	$O(\log(n))$ +	$O(1)$ ++
Hashtabel	n.v.t.	$O(1)$ ++	$O(1)$ ++ Zolang binnen grootte

Bedenk een oplossing

- ◆ Gegeven een groot aantal gegevens (bvb Anna, Bob, Diana, Eddy, ...).
- ◆ Sla ze op in een lijst (grootte mag je zelf bepalen), je mag zelf kiezen op welke positie.
- ◆ Uitdaging: zorg dat je ze *supersnel* kan terugvinden!!



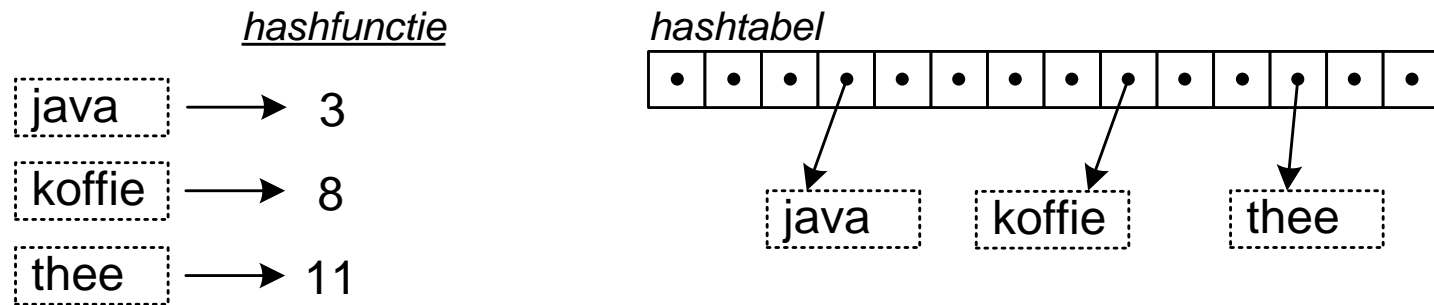
Hashing

- ◆ Probleem: iets terugvinden in een collectie gegevens
 - ✦ Is in feite een **functie**:
 - Input: object
 - Output: plaats in geheugen
 - ✦ Voor arrays, linked lists of bomen doen we dit met het 'doorploeteren' van de datastructuur. We weten niet waar de elementen zich bevinden.
 - Een word beginnend met een 'b' zal zich in het begin van een gesorteerde array bevinden, maar waar precies? Als er veel woorden met een 'a' beginnen, kan dit toch pas ver in de array zijn.
 - ✦ Maar waarom niet:
 - Elk woord een vaste plaats geven
 - Via een echte wiskundige functie de plaats bepalen
- Hashfunctie!

Hashfunctie

◆ $\text{Index} = \text{hashfunctie}(\text{object})$

✦ Bepaalt waar object moet komen in array (hashtabel/hashmap)



✦ $O(1)$ zoektijd, onafhankelijk van de grootte van de array!

◆ Hoe kiezen we de **hashfunctie**?

(a) Hashfunctie gebaseerd op letters

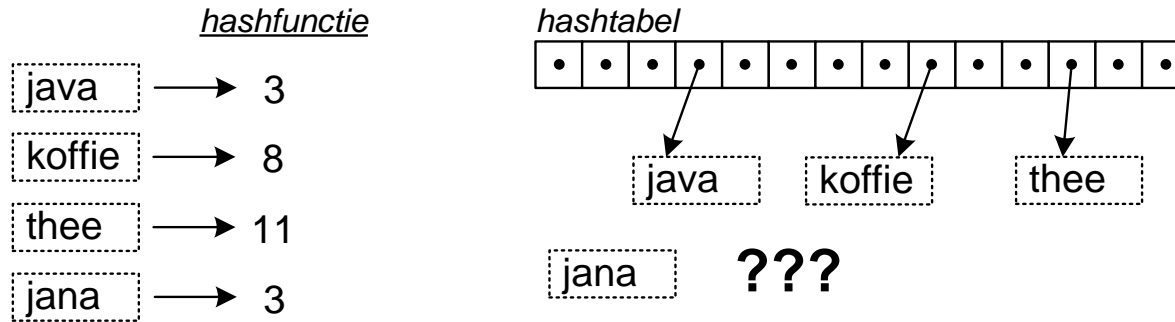
- ◆ Eerste letter nemen van elk woord en geef alfabetische code
 - ✦ `'a' = 0; 'b' = 1; ...`
 - ✦ `Hashwaarde = code(letter) = alfabetische index(letter)`
- ◆ Slechts 26 mogelijkheden
 - ✦ Alle woorden met dezelfde beginletter kunnen we samenzetten (bvb gelinkte lijst, zie verder), maar dan kunnen die niet snel opgezocht worden!
- ◆ *Range van de hashfunctie moet groter zijn dan arraygrootte*

(b) Hashfunctie gebaseerd op meerdere letters

◆ Eerste 2 letters van elk woord

- ✦ $\text{hashwaarde}(\text{woord}) = \text{code}(\text{eerste letter}) * 26 + \text{code}(\text{tweede letter})$
- ✦ "aa" geeft 0
- ✦ "ab" geeft 1
- ✦ "ba" geeft 26
- ✦ "zz" $25*26+25=26^2-1$ = grootste waarde

(c) Enig probleem met hashing: *botsingen*

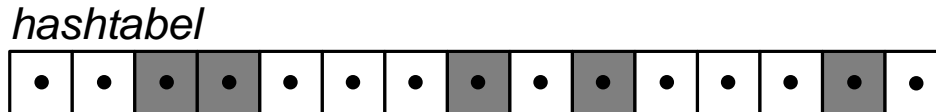


➔ goede hashfunctie maakt deze kans klein

✦ Vb: "java" & "jana" botsen omdat enkel de eerste 2 letters gebruikt worden (\approx modulo operatie)

➔ *beter is om alle letters te laten meetellen*

✦ Ideale hashfunctie: kans op botsing = kans op toevallige botsing = vullingsgraad van array



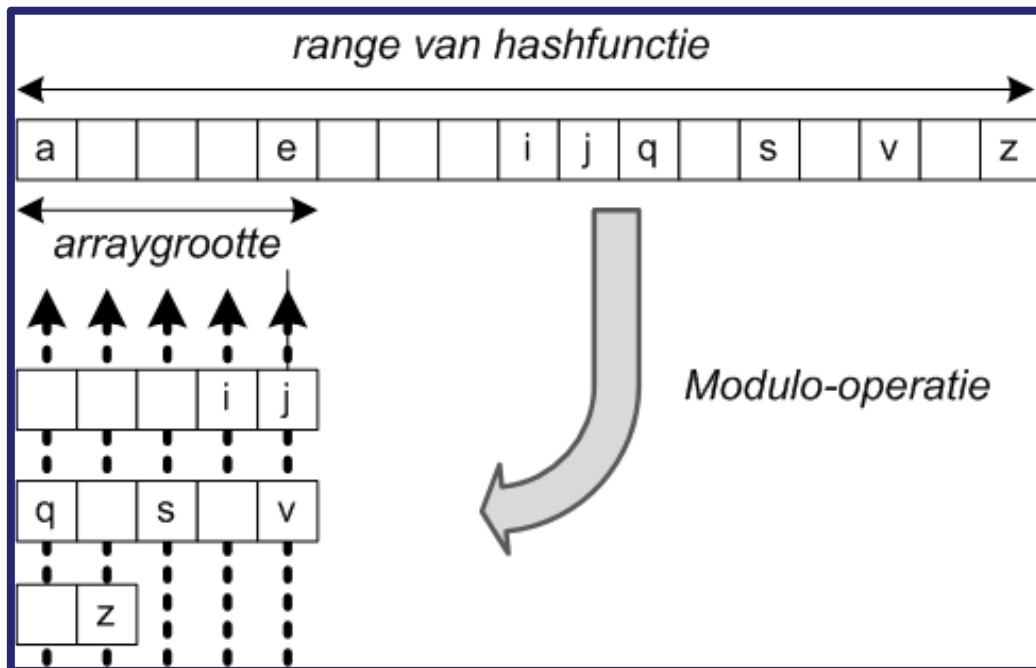
Tabel is voor 30% gevuld (grijze vakken), kans op botsing dus liefst maar 1/3

(d) Alle letters van sleutel (key) moeten meetellen

- ◆ Botsingen als 2 eerste letters hetzelfde zijn
- ◆ *Beter dat alle karakters een **aparte** invloed op de sleutel hebben*
 - ✦ *Stel: $\text{hashwaarde}(\text{woord}) = \text{index}(1^{\text{ste}} \text{ letter}) + \text{index}(2^{\text{de}} \text{ letter})$*
 - ✦ *Dan: 'ab' en 'ba' geven dezelfde hashwaarde*
 - ✦ *Willen we niet => 2^e letter met factor vermenigvuldigen*
- ◆ *Hashwaarde moet groter zijn dan arraygrootte*
- ◆ Hoe? Neem alle letters mee, voor elke letter:
$$\text{hashwaarde}(\text{woord}) = \text{som}[\text{over alle } i\text{'s}] \text{index}(i^{\text{de}} \text{ letter}) * 26^i$$

(e) Hashwaarde binnen arraygrootte

- ◆ Neem maximale hashwaarde groter dan arraygrootte en neem dan de modulo:
 - ✦ $\text{index} = \text{hashwaarde} \bmod \text{arraygrootte}$
- ◆ Range van hashfunctie mag véél groter dan arraygrootte
 - ✦ Hierdoor wordt alles goed gemengd en bekomen we een random gedrag



Er zijn meer dan gemiddeld woorden met een 'a' en 'e'. Met een brede hashfunctie smeren we die uit over meerdere vakjes (in schema aangeduid met slechts 1 vakje).

Na de modulo-operatie komen woorden met een 'q' op dezelfde positie als woorden met een 'a'. Omdat 'q' weinig voorkomt is dit een goede zaak. Er ontstaat een random verdeling.

(f) Neem priemgetal voor arraygrootte

- ◆ Stel: de 8-bitkeys hebben de volgende binaire waarden:
 - ✦ 00010110 (decimaal 22) % 8 = 110 (decimaal 6)
 - ✦ 00000110 (decimaal 6) % 8 = 110 (decimaal 6)
 - ✦ 11011110 (decimaal 222) % 8 = 110 (decimaal 6)
- ◆ Als arraygrootte = 8
 - ✦ => `key % 8` betekent dat je de 3 rechtse bits neemt (onderlijnt hierboven)
 - ✦ => de 3 getallen geven dezelfde waarde en komen op dezelfde plaats in de array zitten: 3-voudige botsing!!
 - ✦ Niet alle bits spelen een rol in de hashfunctie, waardoor meer kans op botsingen!!
- ◆ Dit voorkom je door een priemgetal te nemen voor de grootte:
 - ✦ `Key modulo priemgetal` zal een index opleveren die rekening houdt met alle bits

Uitwerking hashfunctie

- ◆ **Oud-examenvraag:** Leg uit hoe een hashtabel werkt. Wat is een goede hashfunctie? Stel een goede hashfunctie op voor het opslaan van alle Belgische adressen (met straat, huisnummer, bus, postcode en gemeente).
- ✦ Hoe worden botsingen opgelost? Wat zijn de nadelen van hashing?

Oplossing.

- De maximale hashwaarde moet minstens 10 miljoen zijn (het aantal adressen van België), maar mag gerust meer zijn gezien de modulo.
- Als grootte van de array nemen we een priemgetal groter dan 10 miljoen.
- We wensen verschillende hashwaarden voor ongelijke adressen, dus gebruiken we alle onderdelen van het adres in de berekening van de hashwaarde.

Goede hashfunctie (stringcode hebben we hierboven gedefinieerd):

$100 * \text{stringcode}(\text{straat}) + 10 * \text{huisnummer} + \text{postbus} + 10^5 * \text{postcode}$

HashCode: in Java Object

protected [Object](#) [clone](#)() Creates and returns a copy of this object.

boolean [equals](#)([Object](#) obj) Indicates whether some other object is "equal to" this one.

protected void [finalize](#)() Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.

[Class](#)<?> [getClass](#)() Returns the runtime class of this Object.

int [hashCode](#)() Returns a hash code value for the object.

void [notify](#)() Wakes up a single thread that is waiting on this object's monitor.

void [notifyAll](#)() Wakes up all threads that are waiting on this object's monitor.

[String](#) [toString](#)() Returns a string representation of the object.

void [wait](#)() Causes the current thread to wait until another thread invokes the

*Java geeft default hashcode,
overschrijf indien je die wilt verbeteren*

void [wait](#)(long timeout, int nanos) Causes the current thread to wait until another thread invokes the [notify\(\)](#) method or the [notifyAll\(\)](#) method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed.

Hoofdprobleem: botsingen

◆ **Ideaal:** de array voor $p\%$ gevuld

=> kans op botsing ook $p\%$

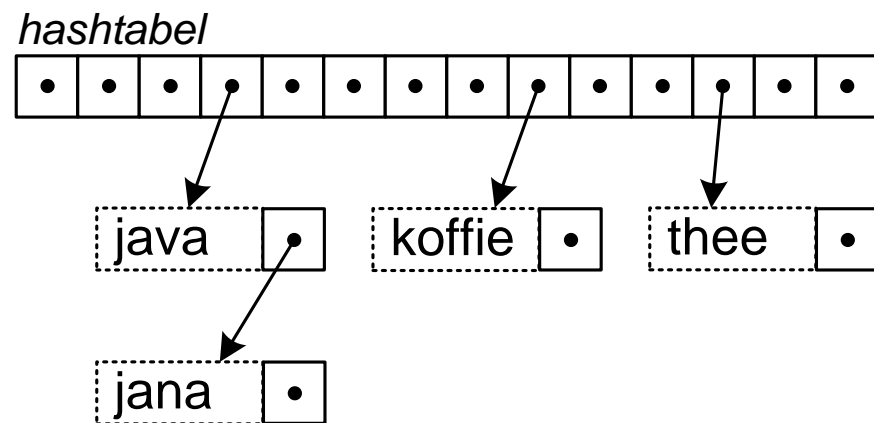
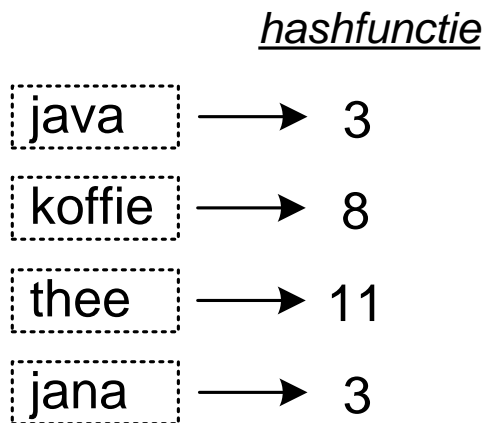
◆ **Perfect hashing** (=geen botsingen) enkel mogelijk als verzameling waarden (sleutels) op voorhand gekend is

✦ Bvb de gereserveerde woorden van java (zoals class, public, static, void, ...)

Botsingen - Oplossing 1: lijst

◆ Alle elementen met zelfde index in lijst zetten

- ✦ Bvb linked list.
- ✦ Echter, opzoeken in gelinkte lijst is traag...



Botsingen - Oplossing 2: alternatieve index berekenen

◆ *lineaire*

✦ $h_i = (\text{hash}(\text{sleutel}) + i) \text{ MOD } n$

✦ *Nadeel*: de bezette plaatsen concentreren zich in blokken

◆ *kwadratische open adressering*

✦ $h_i = (\text{hash}(\text{sleutel}) + i^2) \text{ MOD } n$

Voorbeeld (h_0 is normale hashwaarde)

Key	$h_0 =$ Hash(Key)	$h_1 = h_0 + 1$	$h_2 = h_0 + 2$	$h_3 = h_0 + 3$ (lineair)	$h_2 = h_0 + 2^2$ (kwadratische)
4215	211				
1212	211	212			
7220	213				
8219	211	212	213	214	215

Nadelen hashing

1. Statische karakter van de datastructuur

- Array kan niet uitgebreid worden, want hashfunctie moet dezelfde blijven

2. Elementen staan ongeordend in lijst

- In volgorde printen kan niet

3. Verwijderen van elementen is soms moeilijk

Kan problemen geven bij botsingen

- Linked lists: OK
- Alternatief adres: als 1^e element verwijderd, hoe weten we dat volgende elementen op alternatieve plaats staan?
 - Mogelijke oplossing: verwijderde objecten laten staan maar aanduiden als verwijderd met vlag

Mapimplementaties

◆ `Map<String, String> map = new TreeMap<String, String>();`

Of

◆ `Map<String, String> map = new HashMap<String, String>();`

Ook voor de implementatie van de **Set** interface heb je de keuze uit **Tree** of **Hash**

Hashfuncties in authenticatie en security

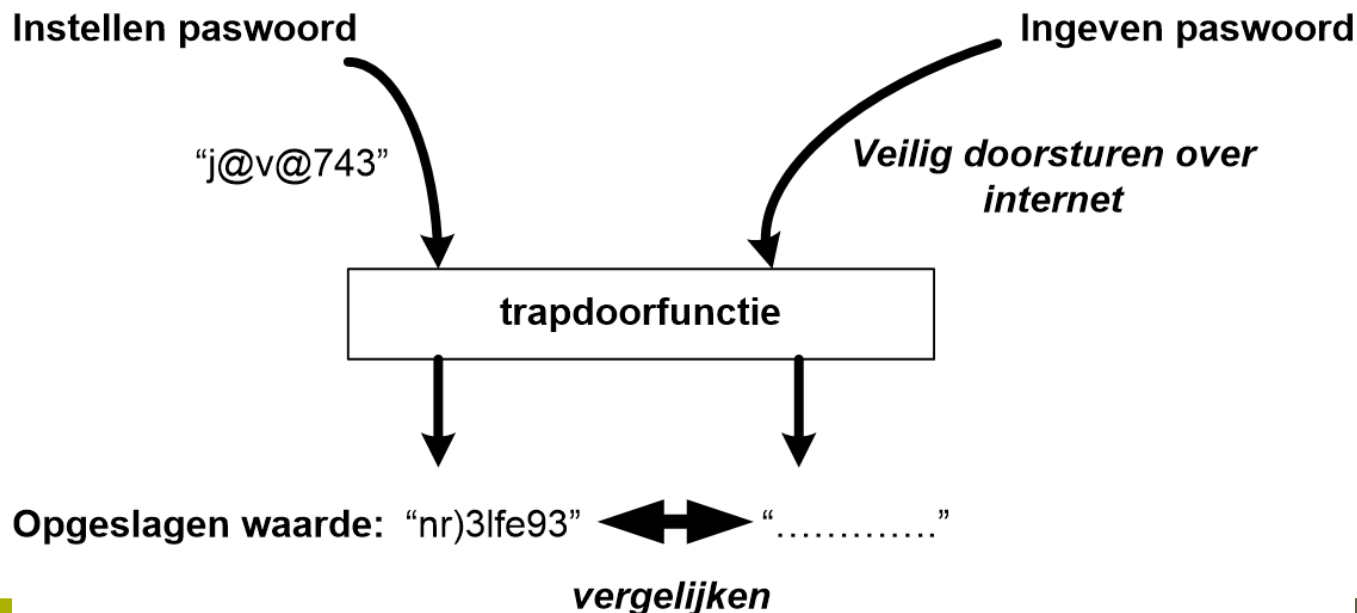
Toepassing (1): Authenticatie van een file

- ◆ Je wilt zeker zijn dat het om een originele file gaat, dat de file niet veranderd is
 - ✦ Checksum = hashfunctie(inhoud file)
 - Alle bytes moeten meetellen in hashfunctie
 - Onvoorspelbaar gedrag: je kan niet achterhalen wat je moet veranderen om een bepaalde checksum te bekomen
 - Elke verandering leidt tot een complete andere hashwaarde (*avalanche effect*)
 - ✦ => complex functie gebruiken, bvb SHA-familie (Secure Hashing Algorithm).
- ◆ Zie highscore server

Hashfuncties in authenticatie en security

Toepassing (2): Niet letterlijk opslaan van paswoorden

- ◆ *Trapdoorfuncties*: functies die in één richting snel uit te rekenen zijn, maar waarvan de **inverse** onbekend is (niet in redelijke tijd uit te rekenen)
- ◆ Paswoord kan men niet achterhalen!





Hoofdstuk 8: Internet & innovatie

Film "The Social Network"

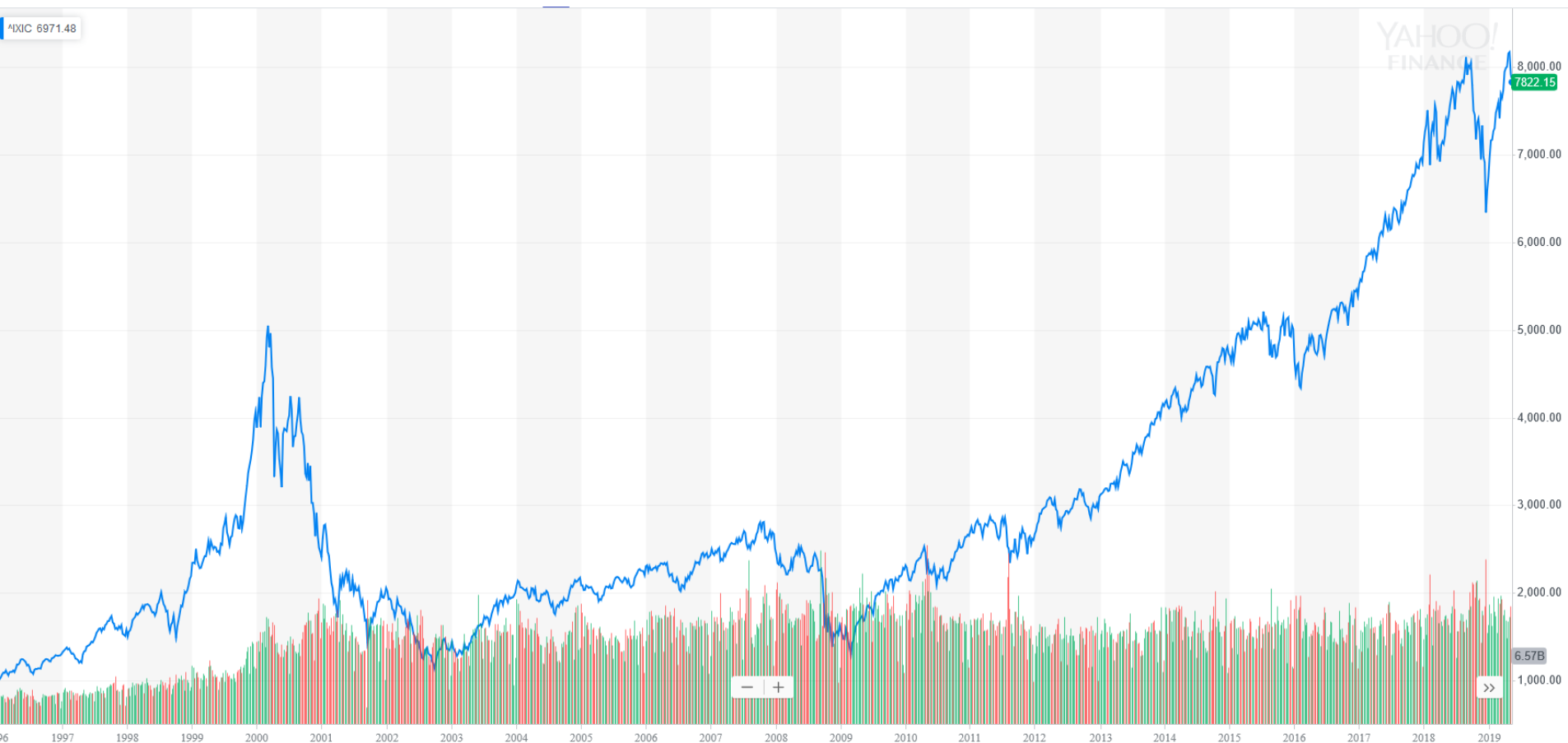
(2010) David Fincher



Over het ontstaan van facebook

Na de internetbubbel

Technologie-index van USA: Nasdaq



Jaren '90 eindigde in de dotcom-crisis

Internet geeft ongekennde nieuwe mogelijkheden

Internet zou de wereld totaal veranderen

Droom spatte uit elkaar...

Na de dotcom-crisis

Investerings vallen (even) stil

Maar: Internet wint meer en meer terrein

En plots... Google begint aan het internet te verdienen

De trein is vertrokken

Komt de droom toch uit?

webtechnologie

◆ Internet 1.0

- ✦ Informatie te bekijken via browser

◆ Internet 2.0

- ✦ Gebruiker interageert en voegt informatie toe

◆ Webservices: informatie wordt ter beschikking gesteld, kan automatisch (door computerprogramma) opgehaald worden ipv. via browser

- ✦ Vb: bustijden, google maps, ...

◆ Internet 3.0: het semantische web

- ✦ Semantiek (betekenis)

De uitvinder van het internet



Tim Berners-Lee

✦ 1989: maakte de eerste webserver (http) en browser (html) samen met de Belg

Robert Cailliau

✦ 1999: dacht verder:

"I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A '**Semantic Web**', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialize."

Door internet: verhoging efficiëntie

- ◆ Sneller en gemakkelijker communiceren
- ◆ Informatie overal aanwezig en toegankelijk
 - ✦ Vroeger: bibliotheek met beperkte info
- ◆ Digitale verwerking
 - ✦ Bvb tax-on-web: belastingen online invullen ipv via formulier die dan ingescand moet worden
- ◆ Webservices: *programma's kunnen info opvragen & gebruiken*
 - ✦ luchtvaartmaatschappijen
 - ✦ ...

Dat internettechnologie heel wat mogelijk maakt is duidelijk, maar technologie is geen garantie op succes...

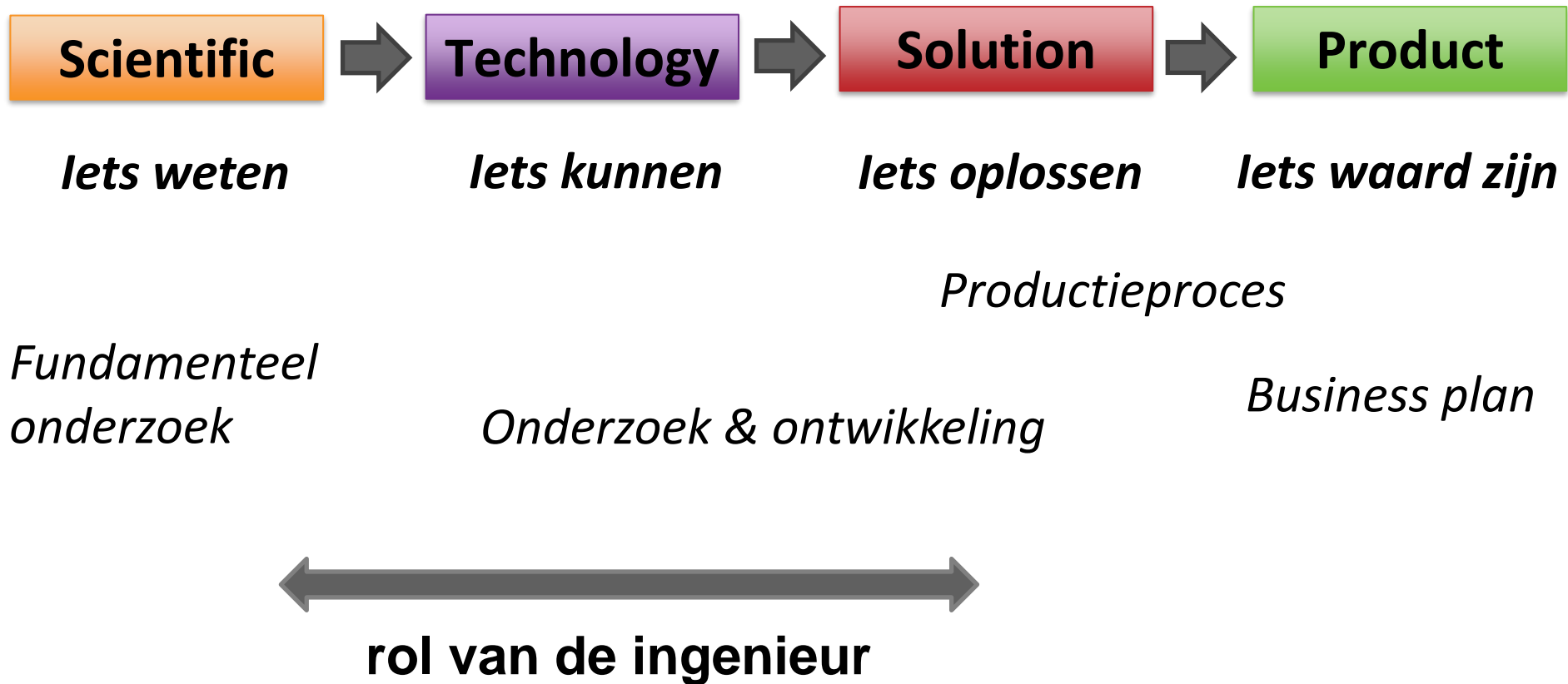
Technologie is niet alles

- ◆ Wat is de bijdrage van technologie tot het succes?
- ◆ Welke technologie is/wordt belangrijk?
- ◆ Onverwachte wendingen...
 - ✦ Sms werd onbedoeld een enorm succes, mms dan weer niet
- ◆ Standaardisatie!
 - ✦ Voorbeeld: Gsm
 - Eenvormig systeem, van operator veranderen is gemakkelijk
 - USA: alle operatoren hebben een ander systeem
 - Gsm is nu dan ook wereldwijd de standaard (behalve in de USA...)

Wat is er nodig voor succes?

Ingenieurs/technenuten hebben het dikwijls moeilijk om dit te begrijpen omdat ze vooral bezig zijn met de technologie

De lange weg naar de markt



Wat is er, naast technologie, nodig voor succes?

(1) Vertrouwen

- ◆ *Vroeger:* securityproblemen
 - ✦ Geen vertrouwen
- ◆ Gebruik VISA, Online banking,
- ◆ Meer en meer online kopen: vertrouwen in tweedehandssites, ...
 - ✦ Je betaalt een onbekende en vertrouwt dat hij het opstuurt!
- ◆ Ons koopgedrag verandert (*maar niet eensklaps en massaal*)
- ◆ Oud voorbeeld: kernenergie
 - ✦ Ingenieur vindt dat de consument maar moest vertrouwen dat alles veilig is. Hij kan niet overweg met 'irrationele' argumenten

(2) Gebruiksvriendelijk

- ◆ Google maps <> Map24 (vroeger de belangrijkste)
 - ✦ Google via handige functionaliteiten veel gebruiksvriendelijker
 - ✦ Ik ben direct overgeschakeld...
- ◆ Google verzekerde dat het resultaat steeds snel ter beschikking was (binnen 1 seconde)
 - ✦ Voorheen moest je al eens lang wachten
- ◆ Apps & muziek
 - ✦ Wat is een 'app' anders dan een softwareprogramma?
 - Afgebakend programma door systeem beheerd
 - Eenvoudig te installeren en up te daten
 - ✦ Itunes store: gemakkelijk muziek kopen

(3) Je moet de grootste zijn

"The winner takes it all"

- ◆ Of ten minste groot genoeg
- ◆ Hoe bereik je kritische massa?
- ◆ Voorbeeld: smart TV
 - ✦ Enkel succesvol met 1 standaard
 - ✦ Vele initiatieven (Apple, Samsung, Google, Microsoft, ...), maar geen duidelijke winnaar. Geen enkele speler gunt de ander de machtspositie!

(4) Het menselijke & sociale aspect

- ◆ Wat zoeken mensen op internet...
 - ✦ 'Contact met andere mensen'
- ◆ Facebook's Zuckerberg: studeerde psychologie (naast computer science)
- ◆ Begrijpen van 'mens'
 - ✦ Apple's Steve Jobs: nadruk op *design*, op *gebruiker* ipv technologie

(5) Goed Business model

Winstgevend zijn op termijn

◆ Google, facebook:

- ✦ via advertenties

- ✦ Hebben informatie over gebruikers => gerichte reclame

◆ Betalende sites (bvb krant): moeilijk

- ✦ We verwachten dat alles gratis is...

Strategie: open versus gesloten

◆ Microsoft's Windows:

- ✦ Open besturingssysteem
- ✦ Iedereen mag er software voor ontwikkelen
 - Geeft andere bedrijven kansen
- ✦ Microsoft concentreerde op besturingssysteem & software
 - niet op hardware en niet op alle software

◆ Apple:

- ✦ Hield en houdt controle over het hele systeem, software & hardware
 - Werkte initieel tegen hun (eind '90 bijna failliet)
 - Via "apps" kan je software aanbieden
- ✦ Pakt nu succesvol uit met 'totaalproducten'
- ✦ Gebruiksgemak, stijl en design steeds prioritair

Analoog: Samsung/Android versus Apple

De grote IT-bedrijven

- ◆ De "Big 5", allen Amerikaans
 - ✦ Je moet de grootste zijn om de winner te zijn. USA is groot, België is klein.
- ◆ Zorgden voor vernieuwing: technologisch, bedrijfsfilosofie en wouden ook spelen op het maatschappelijke (cf Elon Musk)
- ◆ Investeerden veel, hoopten op doorbraken
 - ✦ Google ook in robotica, ...
 - ✦ Maar: De Standaard 2021 01 25 - Google geraakt maar niet op de maan.pdf
- ◆ In het begin: veel sympathie
- ◆ Nu groeiend scepticisme (zeker in USA!)
 - ✦ Privacy
 - ✦ Te groot, te machtig, houden quasi-monopolies, dringen kleine bedrijven uit de markt, betalen nauwelijks belastingen, ...
 - ✦ Zie De Standaard van augustus 2018: "Machtspositie van Google-facebook-apple gevaarlijk"

Belangrijkste IT-bedrijven

Key Statistics:

Revenue

Net Income Avl to Common

= winst/omzet

Market Cap

= beurswaarde/winst

Mei 2015

	Omzet	Winst	Winst-marge	Beurswaarde (koers)	Koers-winst
Microsoft	94,8	20,0	21%	387 (48,7)	19
Apple	212	48	22%	750 (128)	15,6
Google	67	13,8	20%	373 (500)	27
Facebook	13,5	2,8	20%	219 (80)	78

Mei 2016

	Omzet	Winst	Winst-marge	Beurswaarde (koers)	Koers-winst
Microsoft	87	10,5	12%	404 (51,5)	38
Apple	227	50	22%	494 (90)	9,8
Google (Alphabet)	78	17	19%	499 (728)	29
Facebook	19,7	4,6	23%	344 (120)	74

Belangrijkste IT-bedrijven

Key Statistics:

Revenue

Net Income Avl to Common

= winst/omzet

Market Cap

= beurswaarde/winst

Mei 2017

	Omzet	Winst	Winst-marge	Beurswaarde (koers)	Koers-winst (rendement)
Microsoft	87	17,8	20%	521 (67,5)	29 (3,4%)
Apple	220	45,7	21%	783 (150)	17 (5,8%)
Google (Alphabet)	94,8	20,7	22%	643 (919)	31 (3,2%)
Facebook	30,3	11,5	38%	420 (144)	36 (2,7%)
			Mei 2018		
Microsoft	90	21,2	20%	730 (95)	34 (2,9%)
Apple	229	48,3	21%	858 (169)	17,7 (5,6%)
Google (Alphabet)	111	28,8	26%	722 (1016)	25,1 (3,9%)
Facebook	40,6	15,9	39%	503 (174)	31,6 (3,1%)
Amazon	177	3,0	1,7%	767 (1582)	255 (0,4%)

Belangrijkste IT-bedrijven

Vraag: welk aandeel brengt het meeste op?

Key Statistics:

	Omzet	Winst	Winstmarge	Beurswaarde	Koers-winst
	Revenue	Net Income	Avl to Common	= winst/omzet	Market Cap
			Mei 2019		= beurswaarde/winst
					= winst/beurswaarde
Microsoft	122	34,9	28,6%	965 (126)	27 (3,6%)
Apple	258	57,2	22%	878 (190)	15,3 (6,5%)
Google (Alphabet)	142	28,0	19,7%	809 (1164)	28,9 (3,5%)
Facebook	59,0	19,6	33%	532 (186)	27,1 (3,6%)
Amazon	241	12,0	5,0%	921 (1871)	76,8 (1,3%)

	Omzet	Winst	Winstmarge	Beurswaarde	Koers-winst
			Mei 2020	(koers)	(rendement)
Microsoft	138,7	46,2	33%	1390 (183)	30 (3,3%)
Apple	268	57,2	21%	1320 (303)	23,1 (4,3%)
Google (Alphabet)	166	34,5	21%	919 (1369)	26,6 (3,7%)
Facebook	73,3	20,9	28,5%	593 (211)	28,3 (3,5%)
Amazon	296	10,6	3,5%	1180 (2367)	111 (0,9%)

Belangrijke bedrijfsstatistieken

Tussen haakjes de engelse term gebruikt op Yahoo-website onder (Key) Statistics

- ◆ **Omzet** (revenue): totaal bedrag van verkochte producten en diensten
- ◆ **Winst** (Net Income Avl to Common) = Omzet min de kosten
- ◆ **Winstmarge** (%) = Winst / Omzet
- ◆ **Beurskoers** = waarde van 1 aandeel
- ◆ **Beurswaarde** (Market Capitalization) = beurskoers x aantal aandelen
- ◆ **Koers-winstverhouding** = Beurswaarde / Winst
 - ✦ Geeft aan hoeveel jaar het duurt voor je je inzet terug hebt verdient
- ◆ **Rendement** (%) = 1 / Koers-winstverhouding
 - ✦ Winst die je maakt op je aandelen

Koers-winstverhouding

Koers-winstverhouding VS versus eurozone

Evolutie 10 jaar



DS Infografiek | Bron: Thomson Datastream

Facebook

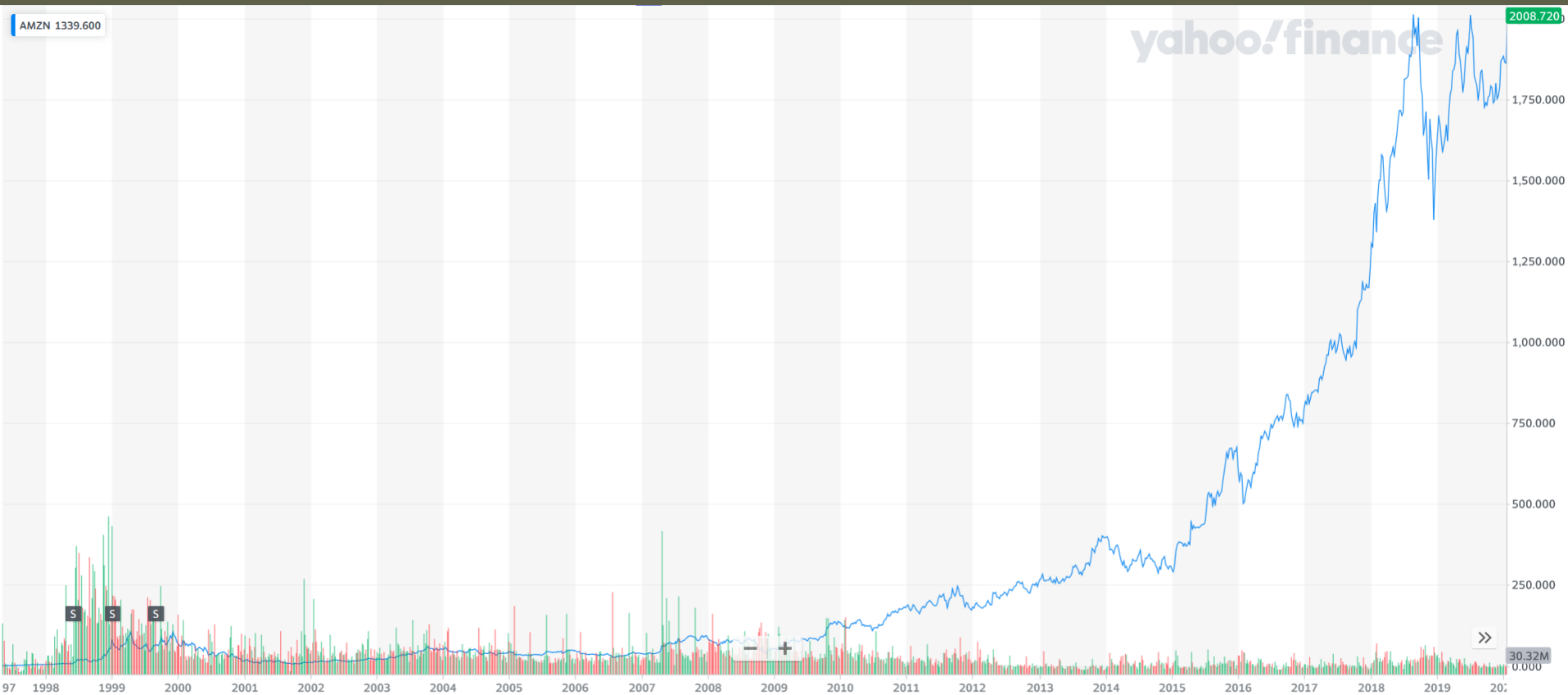
- ◆ Bij beursgang (mei 2012): 100 miljard beurswaarde
 - ✦ 38 dollar initiele prijs aandeel
 - ✦ Steeg onmiddellijk tot 42 dollar
 - ✦ Bleef langer lager, tot goede app voor smartphone ontwikkeld werd
 - ✦ Momenteel grote vragen ivm privacy & inmenging in verkiezingen



Google (Alphabet)

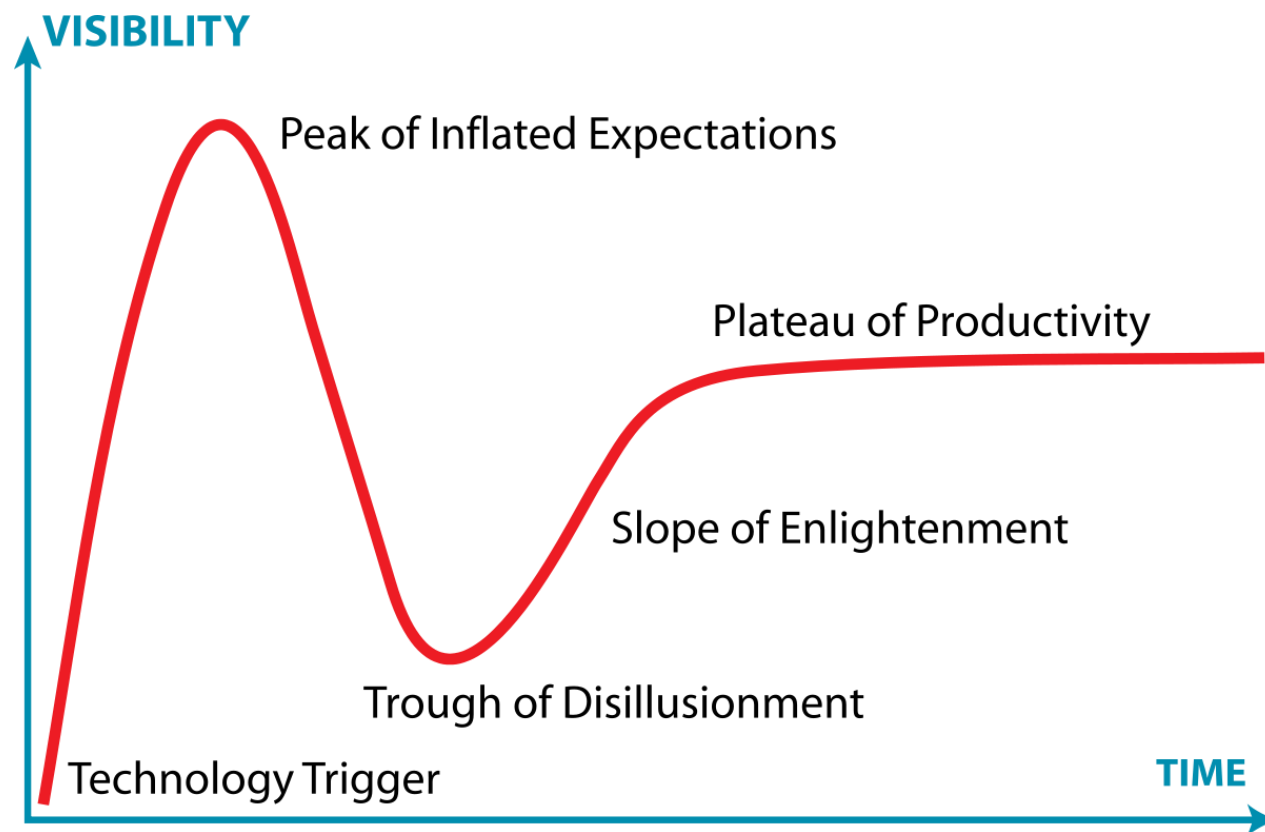


Amazon

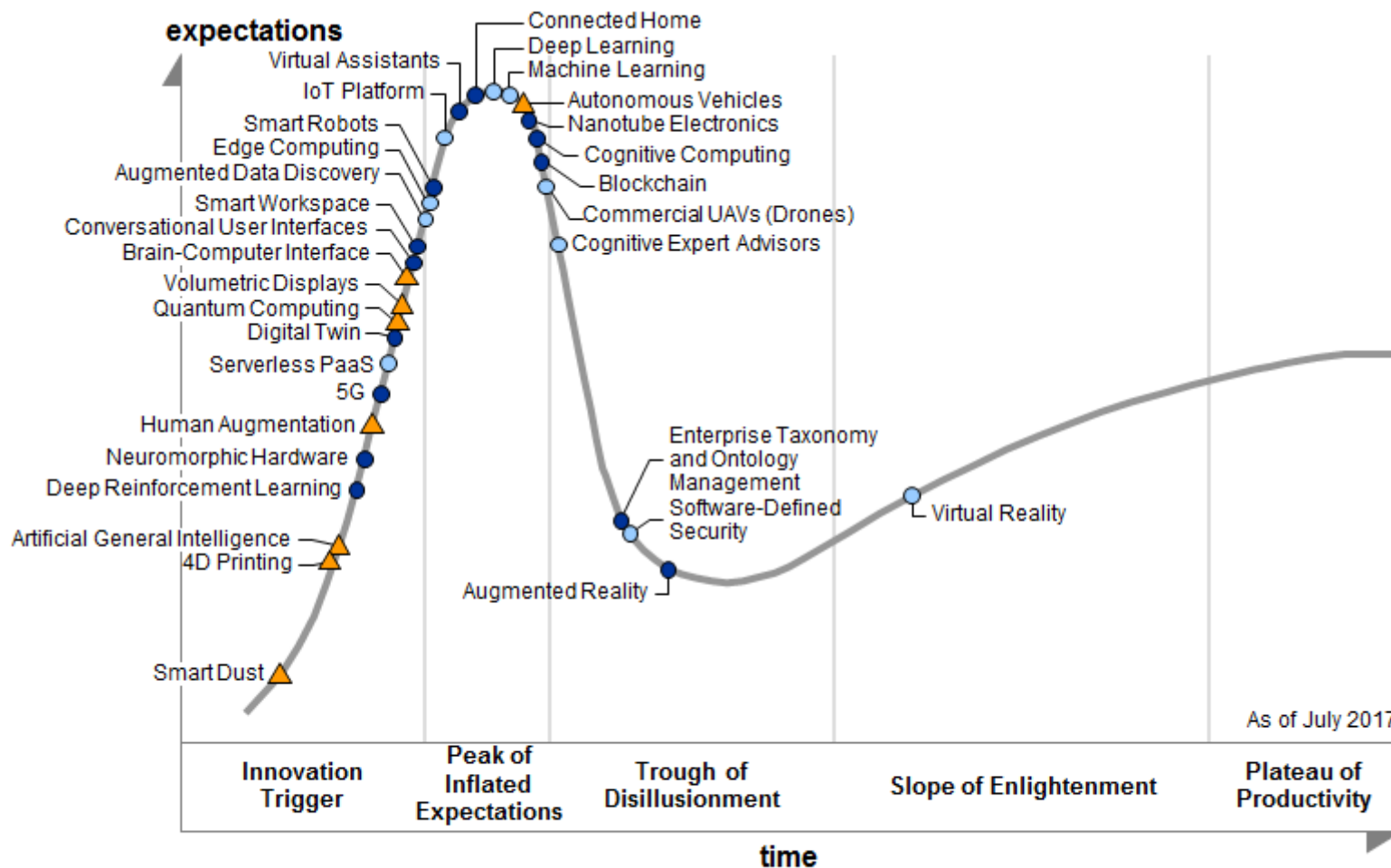


Amazon bestaat al lang. Bleef alle winst investeren.
Bezos dacht op lange termijn en had geduld!!

IT-technologie: ***wat brengt de toekomst?***



2017



2018

expectations



Plateau will be reached:

○ less than 2 years ● 2 to 5 years ● 5 to 10 years ▲ more than 10 years ⊗ obsolete before plateau

Examen

Doel van het vak

Kennis & vaardigheden om informatica te gebruiken als tool

- ◆ Regels kunnen toepassen
- ◆ Redeneren
- ◆ Problemen oplossen
- ◆ Performantie kunnen inschatten

Mondeling examen

- ◆ Schriftelijke voorbereiding (maximaal 2 uur) met mondelinge verdediging (15-20 minuten)
- ◆ **Enkel eerste 20 minuten van de voorbereiding mogen boek en nota's gebruikt worden (open boek)**, daarna is enkel pen en papier toegelaten.
 - ✦ Dit voor 3 vragen van deel I en deel II
 - ✦ Niet van toepassing voor vraag uit deel III & binair rekenen
- ◆ Elke vorm van communicatie of technologische hulpmiddelen (zoals computer) zijn uitgesloten.
- ◆ Voorbeeldexamen op website

**2020-2021-Corona: geen mondelinge verdediging.
Schriftelijk voorbereiden en afgeven. 2uur de tijd.
De rest blijft hetzelfde.**

Ophalen 1^e semester

- ◆ indien je een 7, 8 of 9 behaalde in het eerste semester en je hebt minstens 12/20 voor je mondeling examen
- ◆ Een extra vraag over de materie van 1^e semester waarmee je je punten kan optrekken tot maximaal 10/20
 - ✦ geen specifieke python-vraag, eerder algemene programmatievraag. Je mag je pythonboek gebruiken.

2020-2021-Corona: je krijgt de extra python-vraag op aanvraag.

Deel I: java & object-oriëntatie

Boek dient vooral als hulpmiddel

Wel:

- ◆ de klassikaal-opgeloste oefeningen (p. 62-65)
- ◆ van buiten kennen: **de java spelregels**
 - ✦ Bvb Regels kunnen toepassen op bovenstaande oefeningen
- ◆ Pijlers van object-oriëntatie (p. 7)
 - ✦ Gebruik en nut begrijpen wanneer toegepast op gevallen zoals in de cursus

Deel II: datastructuren & algoritmen

- ◆ Begrijpen, niet kunnen reproduceren
 - ✦ Varianten wel kunnen genereren
- ◆ Op voorbeelden kunnen toepassen
 - ✦ Voorbeeld van lijst, boom, spelsituatie, ...
- ◆ Wat gebeurt er bij kleine varianten?
 - ✦ Loopt het fout? Waar loopt het fout?
- ◆ Optioneel voor goede programmeurs: 5.2.6 generieke code voor de zoekalgoritmen
 - ✦ Je mag dit als vraag kiezen, dan krijg je een andere vraag minder

Voorbeeldvraag:


- ◆ Gegeven: zoekboom
- ◆ Gevraagd: wat doet elk zoekalgoritme? Welk deel van de boom zal het doorzoeken? Hoeveel nodes worden er bekeken en hoeveel bezocht? Wat is je conclusie ivm de snelheid en accuraatheid van de algoritmen
- ◆ Java app online waarmee je het kan testen

Deel III: technologie, historiek en economische aspecten van de IT-wereld.

◆ Aspecten belangrijk voor IT-wereld

- ✦ *Essentie kennen, belangrijke onderscheiden van detail*
- ✦ Eigen mening wordt gewaardeerd (“redelijk eigenzinnig”)
- ✦ Parate kennis: zodat je verdere info kunt kaderen

◆ Te kennen (gesloten boek): hoofdstukken 1 t.e.m. 9

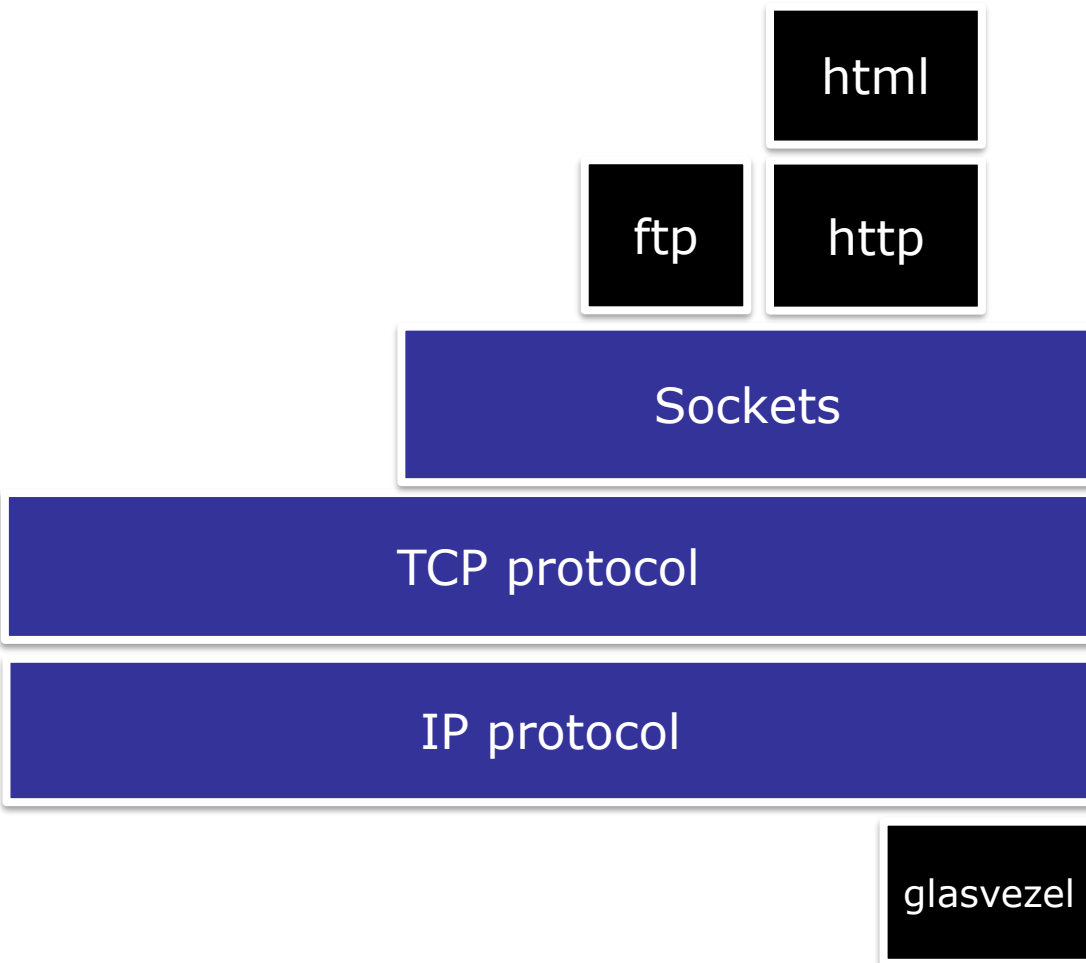
- ✦ **Niet:** namen, geschiedenis, data, wat aangeduid staat met 
- ✦ Hoofdstuk 9: een eigen, onderbouwde mening kunnen formuleren op de volgende vragen: *Wat is volgens u intelligentie? Is de huidige computer al intelligent? Zijn we al goed op weg naar een intelligente computer? Wat verwacht je in de nabije toekomst?*

◆ Extra informatie & eigen inzichten wordt beloond

◆ Ik vraag 5 concepten toe te lichten (wat, waarom, hoe)

- ✦ Bvb: transistor, koers/winst-verhouding, router & modem, chip, digitaal ipv analoog, geheugenhierarchie, process scheduler, ...

Voorbeeldvraag: wat betekenen de volgende begrippen:



Rekenen met bits

- ◆ Op 5 punten (andere 4 vragen zijn op 10)
- ◆ Zie HOC 2
- ◆ Op te lossen zonder boek
- ◆ Voorbeeld:
 - ✦ `int a = 60; int b = 13;` hoeveel is `a ^ b`?
 - ✦ `int x = 85, i = 4;` Schrijf de code om de (i+1)-de bit van x op 1 te zetten
 - ✦ Hoeveel is 0x38D decimaal en hoeveel binair?

Vragen tijdens de blok?

- ◆ Vragenuurtje dag voor het examen
- ◆ Mail je vraag
- ◆ of mail voor een afspraak
- ◆ Of telefonisch
- ◆ jan.lemeire@vub.be of de assistenten
- ◆ 02/629.1679

Einde

Succes verder!

◆ Toegewenst: **discipline** **planning**
doorzicht **doorzettingsvermogen**
concentratie/focus **zelf-coaching**

- ◆ Succes met je carrière, met je dromen
- ◆ En misschien tot later