Kaj so pametne aplikacije?

Pametne aplikacije so ponavadi zelo kompleksne aplikacije, ki vsebujejo kompleksne algoritme, umetno inteligenco. Za take aplikacije je potrebna skrb za infrastrukturo, kjer avtomatiziramo DEVOPS procese(uporaba Dockerja, Kuberneetesa, Jenkinsa). Pri pametnih in naprednih aplikacij je potrebno vzpostaviti CI/CD pipeline, saj omogoča lažjo posodobitev aplikacije.

Analiza videotokov:

Določene pametne aplikacije delajo s podatki, ki so v obliki slik ali videoposnetkov. Če želimo handlat ogromno količino video podatkov, je potrebna vzpostavitev različnih videotokov. Te videe nato obdelujemo s pomočjo določenih metod umetne inteligence. Ti videi se zbirajo v neko podatkovno bazo, ki ji rečemo baza znanja. Tukaj je pomembno da ne shranjujemo vsega ampak samo ključne dele posnetkov, ki imajo konkreten vpliv na metode umetne inteligence.

Kaj je megla v računalnštvu?

Megla Fog ali Edge computing je trenutno zelo nov termin. V nekih aplikacij bi lahko shranjevali vse videoposnetke, slike, neke druge kompleksne podatke...Obstajajo rešitve za to. Namreč Ne shranjujemo nepomembnih podatkov, saj moramo hraniti podatke trajnostjo z namenom da ne porabimo neskončno veliko pomnilnika. Torej delamo s podatki tako da je to trajnostno in energetsko učinkovito. Edge computing je mišljeno kot: Nemoremo zdej 5TB podatku konstanto pošiljat iiz enega podatkovnega centra v drug podatkovni center. Ztu je fajn da upostevamo une zgornje stvari. Aja pa fajn je da vse zoptimizramo z DEVOPS procesi. Tuki kar se tiče samih aplikacij. Potrebno je zbrati tapravo arhitekturo...npr Mikrostoritve. Usaka manjša »aplikacija« sestavlja večjo aplikacijo. Ti delčki so postavljeni vsak v svoj Docker kontejner. Te kontejnerje sproti ugašamo ter prižigamo glede na uporabnikove potrebe. To je ful fajn kr nerabi bit celotna storitev bit v pripravljenosti cel čas.

Katera področja pametne specializacije poznamo?

Zdravje(imamo inteligentno infrastrukturo)

Gospodarstvo(opravljanje naloge finančnih svetovalcev)

Mobilnost(samo vozeča vozila Tesla)

Javne storitve(javni avtobusi ki delujejo brez voznika)

Energija in trajnost(uporaba naprednih algoritmov – optimizacija pametnih ur)

Novice,mediji ter zabava(Siri, Alexa)

Industrija(Samovozeči vozički, roboti)

Turizem(umetna inteligenca, ki priporča posameznim uporabnikom lokacije, ki naj bi jihgž najbolj zanimale)

Izobraževanje(AI iz dolgih učbenikov ustvari vprašanja)

Naštej značilnosti Von Neumannove arhitektur:

Ustvaril jo je John Von Neumann. Pojavila med letom 1940 in 1950. Začetki same arhitekturo segajo že v čas Charlsea Babbageja(oče računalništva), ki je razvil diferenčni stroj. Iz tega stroja se je razvil Analitično stroj, ki je poleg dosedanjih polinomskih funkcij lahko izvajal tudi druge računske operacije. Vseboval je že osnovne enote Von Neumannove arhitekture: mlin (ALE), shramba (pomnilnik), operator (procesor), izhodna enota(kartice). Sama Von Neumannova arhitektura vsebuje CPE, pomnilnik, V/I enote. Arhitektura se je ohranila vse do danes. Slaba lastnost je ozko grlo(CPE ful hitr, pomnilnik počasen) – reševanje z predpomnilnikom.

Prva programerka:

Ada Bayron Lovelace – izračunala Berullijeva števila. Prva, ki se je zavdela da lahko ss temi stroji rešuješ zelo kompleksne probleme.

Generacije računalnikov – naštej pa opiši:

1. Generacija: elektronke, luknjaste kartice
2. Prvi tranzistorji, releji, prvi diski – prva visokonivojska jezika(FORTRAN,COBOL)
3. Integrirana vezja, manjše velikost računalnikov
4. Čipi, pojav prvih osebnih računalnikov (ZX Spectrum, Commodore 64)
5. Superračunalniki, hitri procesorji, umetna inteligenca visokoresolucijska grafika

Opiši Kako deluje RAM(torej kako deluje MDR,MAR register)

Ram je random access memmory. Predstavljamo si ga lahko kot velika knjižna polica, vsak predal ima svoj naslov. Dostop do nekega naključnega naslova je vedno enak. Podatki se ob izgubu elektrike ne ohranijo. Celice so porazdeljene v vrstice(1 pomnilniška beseda oziroma odvison od arhitekture) ter stolpce.

Branje podatkov iz RAM:

1. MAR register vzame naš željen naslov
2. MAR dekodira ta naslov
3. Iz tega naslova prepišemo vrednost v MDR register

Zapisovanje podatkov v RAM:

1. MAR pridobi naš željen naslov
2. Procesor pošlje našo vrednost v MDR
3. MAR dekodira naslov
4. MAR zapiše vrednost iz MDR-ja na MAR.

Kako je sestavljen ALE?

To je aritmetična logična enota, ki skrbi za izvajanje računskih ter logičnih operacij. Je osnovna sestavina Von Neumannove arhitekture.Imamo povezave med dvema ali več registroma. Obe vrednosti prebere neko »računsko« vezje ki poskrbi za novo vrednost. Imamo različna vezja npr. Vezja za seštevanje, odštevanje, deljenje, množenje, vezje za primerjavo dveh n-bitnih števil itd itd. Tukaj je zelo pomembna uporaba raznih dekodirnikov, ki izberejo tapravo pomnilniško vrednost ali pa izbirnikov. Tuki pri ALE lahko opišeš še mal vezje za seštevanje(npr. Maš vezje za seštevanje in maš posebi vezje za Carry).

Opiši krmilno enoto:

Prebere ukaz iz pomnilnika in ga dekodira, Ugotovi kaj naj se s tem zgodi. Potem ta ukaz izvede tako da ukaz pošlje ALE enoti ali pomnilniku. Imamo tudi programski števec, ki vsebuja naslov naslednje ukaza. Potem pa imamo še Ukazni register, ki vsebuje kodo trenutnega ukaza.Nakoncu mamo še vezje za dekodiranje tega ukaza.

Opiši Von Neumannov Cikel:

To je cikel naloži-dekodiraj-izvedi: Torej najprej naloži ukaz iz pomnilnika, nato se ukaz dekodira pol ga pa še izvedemo.(Tuki lahko opišemo delovanje MDR,MAR, programske števca ter ukaznega števca)

Št bitov, ki jih imajo zastavice GT,EQ,LT,NEQ...

Baje da so veliki 4 bite sam nism 100%

Kako bi lahko nek sistem pohitrili?

Uporabili bi paralelno procesiranje(Če ne zmoreš zgraditi naprave, ki bi delala dvakrat hitreje, zgradi napravo, ki bo naredila dve stvari hkrati. Rezultat bo identičen).Pznamo dva pristopa: SIMD pa MIMD. Aja pa uporabili bi agloritme za paralelno procesiranje

SIMD: Imamo kontrolno enoto, ki nadzoruje več ALE enot. Vsaka ALE enota ima svoj pomnilnik in dela nad svojimi podatki.

MIMD: To je pa nekakšno medpovezavno omrežje, kjer imamo ogromno nekih procesorjev(grid computing) in vsak procesor ima svoj pomnilnik. Računalnik izvrši več ukazov nad različnimi podatki. Tukaj je problem komunikacije med posameznimi enotami saj lahko upočasnijo delovanje.

Kaj pravi Moorov zakon?

Št tranzistorjev se vsakih 18-24 mesecev podvoji, pri tem pa ostane cena procesorja nespremenja.

Kako so števila, znaki, zvok, slike predstavljeni v računalniku?

Podatki se v naravi nahajajo v analogni obliki(torej mamo neskončno mnogo vrednosti). Te podatke je potrebno spremeniti v digitalno(diskretno) obliko podatkov. Nek signal je najprej potrebno vzorčiti, nato pa še kvantizirati. Več bitov ko imamo na voljo za vzorčenje, natančenje bomo predstavili ta podatek(npr. zvok). Kadar predstavljamo informacije je predvsem pomembno da jih predstavimo jasno zanesljivo ter nedvoumno. Imamo zunanjo predstavitev – npr. Črke na monitorju, ter notranjo predstavitev – npr. Za usako ascii vrednost imamo svojo binarno št. Ki ga zna naš sistem prebrati. Slika je predstavljena ponavadi z RGB notacijo. Za rdečo barvo imamo 8 bitov, za zeleno imamo 8 bitov ter za modro imamo 8 bitov. Glede na posamične teže teh barv lahko monitor izriše piksle(tu so pike na zaslone).

Kako so Boolovi izrazi povezani z računalniškimi vezji?

Računske operacije lahko izvajajmo s pomočjo Boolove logike. Boolova logika opisuje vse vrednosti z 0 ali 1. Poznamo tudi Boolove operatorje(AND,OR,XOR,NAND,NOR). Iz teh operatorjev lahko sestavimo pravilnostno tabelo. Najlažje da kar uporabimo DNO(Disjunktivno normalno obliko). Iz tega pod izrazov lahko zgradimo vezje s pomočjo NOT in AND vrat. Vse pod izraze oziroma vsa pod vezje potem povežemo z OR vrati. Vrednost, ki pride skozi izhod je naša željena vrednost glede na kombinacijo vhodnih vrednosti izjavnih spremenljivk. Tole zgorej basiclly opisuje kako mi zgradimo neko digitalno vezje.

Vrata NOT:

Mamo 1 tranzistor

Vrata AND:

Mamo dva tranzistorja, ki sta vezana zaporedno(torej če se pr obeh una nožica odpre pol gre elektrika čez in pride logična enka)

Vrata NAND:

Isto kot AND sam da vsebuje še Boolova vrata za NOT, torej da negira vrednost.

Vrata OR:

Dva tranzistorja vezana vzporedno(torej če se 1 vrata odprejo bos tekla elektrika in bo logična enka)

Vrata NOR:

Isto kot OR sam še NOT vezje imamo, ki spremeni pol logično vrednost.

Opiši kako zgradimo in kako deluje vezje za primerjanje?

Imamo dve spremenljivki npr A pa B. Napišemo pravilnostno tabelo za ekvivalenco: A ⬄ b. Iz tega zapišemo DNO obliko te tabele. Vse pod izraze realiziramo z AND in NOT vrati, vsa ta pod vezja pa povežemo z OR vrati. Nakonc dobimo 1 če sta bita enaka, oziroma dobimo 0 če sta bita različna.

Opiši razlike med Izbirnikom in Dkodirnikom:

Izbirnik izbere enega izmed mnogoh vhodov kot to določajo izbirne linije.(Primer multiplekserja)

Dekodirnik. Imamo veliko vhodov, ampak izbral bo le tisti vhod, ki ustreza vzorcu na vhodnih linijah.

Katere so glavne sistemske naloge?

Naloga sistemske programske opreme je da je vmesnik med človekom in strojne opreme. Glavne naloge so predvsem skritje kompleksne notranje strukture Von-Neumannove ahritekture, predstavitev informacij uporabniku na razumljiv način ter zagotoviti varno okolje za delovanje računalnika.

Prednosti zbirnega jezika pred strojnim jezikom?

Strojni jezik so 1 pa 0, ki ga razume računalnik. Mi ne programiramo v strojnem jeziku, zato ker je človeku težko razumljiv(npr. Človek ne bo vedel kaj pomeni npr koda 1001101011), zato uporabimo zbirni jezik.Nasploh je zbirni jezik bolj jasen berljiv ter vzdrževalen. Zbirni jezik vsebuje ukaze za osnovne matematične operacije, nalaganja podatkov iz pomnilnika ter pisanje novih podatkov v pomnilnik. Ker je zbirni jezik razumljiv, lahko v njem optimiziramo našo programsko kodo. Iz zbirnika se koda ukazov prevede v strojni jezik, ki ga razume računalnik.

Kaj so naloge operacijskih sistemov?

1. Zagotavljanje delovanje uporabniškega vmesnika
2. Dostop do Sistema in datotek
3. Varnost in zaščita Sistema
4. Razporejanje procesorskih virov med različne programe
5. Procesiranje V/I enot
6. Reševanje neskočnih zank
7. Premikanje programov v in iz pomnilnika
8. …

Generacije operacijskih sistemov?

Prva generacija: programerji delali neposredno z mašino(sploh še ni blu OS)

Druga generacija: paketni operacijski sistemi, ki je vseboval jezik za nadzor opravil

Tretja generacija: večopravilni operacijski sistemi(več programov naloženih v pomnilnikih, princip centralnega računalnika in terminalov)

Četrta generacija: podpora lokalnih ter omrežnih storitev, pojava grafičnih uporabniških vmesnikov(Realno-časovni operacijski sistemi)

Tole spodej lahko tut kej poveš:

Najprej smo imeli operacijske sisteme, ki niso imeli grafičnega vmesnika. Imeli so le terminal(ukazno vrstico), iz kjer smo vžigali ter ugašali programe. To je bil npr MS DOS. Niso bili narejeni za multitasking(odprt le 1 program hkrati). Potem pa so se pojavili naprednejši operacijski sistemi. Npr Windows, Linux(osnova je Unix), ki vsebujejo že grafični vmesnik in lahko vzdržujejo delovanje več programov hkrati.

Kaj je to formalni jezik?

Formalni jezik je množica besed končne dolžine, ki so sestavljene iz simbolov neke izbrane končne množice simbolovm, ki ji rečemo abeceda.

Kaj je gramatika?

Gramatika so pravila zapisovanje nekega določenega jezika. Lahko npr. Sestavimo gramatiko za nek programski jezik. Praviom pravimo produkcije, vsaka produkcija ima levo in desno stran. Iz neke leve in desne produkcije lahko sestavljamo stavke in iz teh stavkov lahko sestavljamo nove daljše stavke s pomočjo nadalnjih produkcij.Po tej gramatiki mora nato prevajalnik znati prevesti program, ki smo ga zapisali. Poznamo več vrst gramatik:

Gramatike brez omejitev: na levi in desni strain produkcije povsem poljubni simboli

Konteksno-odvisno gramatike: Niz na desni strain produkcije mora biti vsaj toliko dolg kot je niz dolg na levi strain produkcije.

Konteksno neodvisne gramatike: simboli na levi in desni strani produkcije nimajo nobenega vpliva iz strani “konteksta”.

Linearne gramatike: neb vedu kaj je fora sam to je smzdi regularni jezik.

Kakšne so povezave oziroma razlike med jeziki in stroji?

Jeziki se zelo povezujejo z avtomati. Prvo je potrebn razčistiti to, da imamo jezike, ki imajo različno moč izražanja. Imamo regularni jezik, ki zna poizskati ali so določeni pod nizi deli večjih nizov. Potem imamo konteksno-neodvisni jezik. Uporaba produkcij za dva niza ni odvisna od konteksta. Potem so konteksno-odvisni jezik, kjer naredimo produkcijo dveh nizov. Mi lahko glede na nek kontekst oziroma pravilo združimo ta dva niza v neko novo produkcijo novega niza. Tukaj pridejo v igro tudi stroji aka avtomati. Končni Avtomat je model, ki ima končno št stanj , eno izmed stanj ima začetno stanje, nato pa v končenm št koraku odgovori ali nek podniz pripada večjemu nizu. Skladovni Avtomat je model, ki ima poleg končenga števila stanj še neskončno dol sklad, kamor shranjuje začasne “izračunane” simbole oziroma pod nize. To si lahko predstavljamo kot pomožne račune pri matematiki. Deluje podobno kot končni Avtomat le da je naslednje stanje ODVISNO OD SIMBOLA NA VRHU SKLADA. Torej glede na vrsto avtomata lahko bi “razumemo” določeno vrsto jezika. Glej naslednje vprašanje za lažje razumevanje.

Opiši hierarihijo po Chomskemu:

Turingov jezik:

1. Gramatike brez omejitev
2. Turingovi stroji
3. Z njim lahko rešimo vsak rešljiv problem

Konteksno odvisni jeziki:

1. Konteksno odvisne gramatike
2. Linearno omejeni avtomati

Konteksno neodvisni jeziki:

1. Konteksno neodvisne gramatike
2. Skladovni avtomati

Regularni jeziki:

1. Linearne gramatike
2. Končni avtomati

Opiši delovanje Turingovega stroja:

Turingov stroj vsebuje Turingov jezik, ki ga je ustvaril Alan Turing. Ta stroj deluje tako da imamo bralno-pisalno glavo ter neskončno dolg trak v katerih je zapisano neko stanje. Potem stroju podamo več različnih kombinacij branja stanj npr. (1,0,1,2,right) – to je eno stanje in pomeni: če je stanje na glavi 1 in če je na trenutnem delu traka zapisana 0, potem zapiši novo vrednost 1, ter stanje glave postavi na vrednost 2. Glava naj se pomakne za en korak v desno.

Turingov stroj je še posebej zanimiv, ker lahko z njim rešimo vsaj rešljiv problem. Npr. Če v Javi zapišemo nek program, ki je dolg 500 vrstic, bomo ta isti program lahko zapisali na Turingov stroj(sam bi vseboval npr. Miljon ukazov).

Kaj je nerešljiv problem? Povej kak primer.

Nerešljiv primer je je problem, ki ga ne moremo rešiti z Turingovim strojem. Primer tega je primer ustavljanja neskončne zanke(npr. Nemoremo dokazati da se bo program While True ustavil v nekem končem času). Drugi primeri nerešljivih primerov: “Ali je neka BNF gramatika dvoumna?”, “Ali neki dve BNF gramatiki proizvedeta enak jezik?”, “Ali se bo nek program ustavil ne glede na njegov vhod?”

Kaj je Church-Turingova teza?

Pravi: Če obstaja algoritem, ki reši neko nalogo z manipulacijo nekih simboličnih podatkov, potem obstaja Turingov stroj, ki zna rešiti isto nalogo. Ta teza sicer ni dokazana ampak skoraj drži.

Kdaj je nek problem nerešljiv?

Ponavadi so to problem, ki so eksponentni – nepolinomski algoritmi. Varnost računalniških sistemov npr. Temeljijo na temu da so v eno smer polinomski(izračunljivi) ter v drugo smer neizračunljivi. Tuki pride v poštev tut Fermatov izrek, ki ga Vlado učash upraša. Fermatov izrek se nanaša na ostanke pri deljenju s praštevili. Če imamo npr. Neko funkcijo na nek k eksponent in če je k večji kot osnova, potem je ta problem neizračunljiv in bi porabili računalniki miljardo let, da bi prišli do rešitve(neki u tem smislu nevem če je čist prou tule)

Kaj je BNF?

BNF je Backus – Naurova-Forma -> je notacija za konteksno-neodvisne jezike po Chomskemu.

Kako lahko razdelimo jezike?

Poznamo naravne jezike(angleščina, slovenščina, makedonščina…), ki nimajo povezave z strojem(Stroj jih zelo težko razume). Če želimo da stroj razume nek naravni jezik potem je potrebna uporaba hevrističnih algoritmov(algoritmi umetne intelligence, ki so nedeterministični po naravi). Potem je psevdokoda, ki je nekaj vmes med naravnim in programskim jezikom. Je zelo uporabna saj je elegantna, razumljiva vsakemu programerju, na učinkovit način opisuje potek oziroma korake algoritma. Potem imamo programske jezike ki so del visokonivojskih jezikov(C++, Python, Java). Potem je zbirni jezik, ki je povezan s hardewarom. Nakoncu je še strojni jezik, ki ga razume le stroj.

Kako deluje prevajalnik?

Prevajanje lahko razdelimo na 4 faze:

1. LEKSIKALNA ANALIZA. Združevanje posameznih znakov v lekseme(leksemi so neke besede ali pa stavki, ki imajo nek pomen v naši gramatiki programskega jezika).
2. SINTAKSNA ANALIZA. Posamične lekseme združujemo v posamične stavke. Iz the stavkov gradimo eno veliko drevesno strukturo. Naloga te analize je da preveri, ali določeni deli drevesne structure pripadajo naši gramatiki. Če je npr. Stavek While integer I print “Hello”, mora ta korak ugotoviti, da je sintaksa narobe zgrajena.
3. SEMANTIČNA ANALIZA: Preverjamo pomen določenih stavkov. Npr. Mi smo definirali int x = “Hello World”. No ta korak more ugotovit da lahko za x assignamo samo integer in ne stringov. Al pa npr. Da ne delamo operacijo vsote and objekti, ker npr. To ne obstaja v nekem našem jeziku
4. GENERIRANJE KODE: Iz naše preverjene drevesne structure trebe generirati kodo v zbirnem jeziku. Tuki nastopajo tudi različne optimizacije(optimiziramo prevajalnik tako da se nek while zanka prevede v nek preprost ukaz v zbirnem jeziku.), lahko pa optimiziramo naš algoritem(Zboljšamo časovno, prostorsko kompleksnost).

FORTRAN VS COBOL? Vlado je type of guy k se lahk zmisl kej tazga

Fortran je bil narejen za numerično izračunavanje, zahtevne izračune. Vseboval je GO TO stavek. COBOL je pa jezik, čigar namen je bil poudarek na poslovnih aplikacijah. Še danes deluje ogromno kode v obeh jezikih

Java vs Python?

Java se prevede v btye kodo(uporablja prevajalnik), je cross platform kompatibilna, Objektno orientiran jezik. Spremenljivkam je potrebno določiti tip.

Python uporablja interpreter(prevaja vrstico za vrstico), je počasnejši, uporaba za Backend Web development, AI, Machine learning. Aja spremenljivke nimajo tipov.

Kako bi lahko rezdelili programske jezike?

Ja poznamo splošno namenske jezike(C++, Python, Java..) in pa jeziki ki so bili razviti za točno določene naloge.(HTML,XML,OWL,SQL). Jezike lahko razdelimo tudi glede na način programiranja oziroma paradigm: funkcijsko, logično, paralelno…Prolog je za logično programiranje.

Kaj je Algoritem? (Definicija)

Algoritem je popolnoma urejeno zaporedje nedvoumnih ukazov, ki se izvedejo v končnem času in proizvedejo nek rezultat.

Naštej attribute algoritma: (lastnosti)

Hiter, nedvoumen, robusten, eleganten, razumljiv, končen(se ustavi pač da nepride do neskončnih zank).

Kaj je računalništvo?

Računalništvo je veda o načrtovanju in razvoju algoritmov za reševanje množice pomembnih problemov. Poleg tega je računalništvo tudi načrtovanje ter gradnja računalniških sistemov, načrtovanje razvoja programskih jezikov…

Katere operacije se morajo izvesti v algoritmu? Stupid question I know

Računske(iposamezni numerični izračun), nadzorne operacije(vejitve), iteracije, vhod(dobi podatke za procesirat), izhod(posreduje podatke izven algoritma).

Primerjava med zaporednim iskanje ter binarnim?

Zaporedno O(n), binarno 0(log n). Pr binarnem rabmo prvu sortirat podatke. Za sortiranje uporabmo algoritme kot so Quick sort, Merge Sort…itd

Atributi algoritmov(To je ze zgorej napisan ma bo senkrat)

Pravilnost, elegance, učinkovitost, razumljivost

Kako lahko izboljšamo nek algoritem?

Izboljšamo mu časovno kompleknost, tako da se izogibamo potenčni časovni kompleknosti ali pa kaki kvadratni časovni kompleknosti(včasih nimamo druge izbire). Pomembna je tut prostorska časovna kompleknost(uporaba pravih podatkovnih tipov, ki ne ponucajo veliko prostora).Zadnje čase se pojavlja tudi omrežna kompleknost(npr. Da je infrastruktura narjena učinkovito, ko pošiljamo podatke iz enga konca svet na drug)

Katere rede časovne kompleknosti poznas?

Linearna O(n), kvadratna O(n2), kubična O(n3), logaritmična O(log n), linearitmično O(n \* log n), eksponento O(2n), Polinomsko O(nP), p > 0 (tuki se pojavijo kšn šah in ostali nerešljivi problemi), konstantna O(1).

Kaj je fora približnih algoritmov?

Problem bi zahteval slabo časovno kompleknost(npr. Polinomsko časovno kompleknost), zato raje napišemo Hevristični algoritem, ki reši problem “približno” oziroma zadovoljivo.(Hamiltonov cikel, Polnjenje nahrbtnikov, šah…)

Kaj je podatek?

Podatek je neko zabeleženo dejstvo. Sam podatek nima pomena. Je surovina, ki jo predelujejo informacijski sistemi.

Kaj je informacija?

Informacija je logični skupek nekih podatkov, ki prejemniku predstavljajo nek pomen. Informacije lahko prejemniku povečajo znanje.

Katere informacije so kvalitetne?

Tiste informacije, ki so točne, zanesljive, preverljive, varne, dostopne, relevantne.

Kaj je znanje?

O znanju govorimo takrat, ko uporabnik poveže informacije z svojimi izkušnjami, in ga pretvori v obliko, ki je primerna za uporabo.

Kaj je semantični splet?

Semantični splet je ubistvu znanost, ki govori o pomenu.

Kaj je ontologija?

Ontologija je filozofska veda, ki se ukvarja s predstavitvjo ter opisovanjem lastnosti nekega predmeta ter njihovim medsebojnim povezovanjem v resničnostnem svetu. Poznamo OWL (Web Onthology Language), v katerem so stvari pomensko opisane ter organizirane po kategorijah.

Katere poizvedovalne jezike poznaš? Neumnu uprašajne

SQL, Google, Bing, TQL…

Kakšne nevarnosti obstajajo za podatke?

Izguba, Nedostopnost, neželjen dostop, nepooblaščeno spreminjanje podatkov.

Kaj so dokumenti?

Dokumenti so podatki s posebnim statusom. Imajo lahko določeno pravno veljavo, imeti morajo določene lastnosti: vidike celovitosti podatkov.

Vidiki celovitosti podatkov:

Tajnost: določeni dokumenti so dostopni samo pooblaščenim osebam

Trajnost: Podatki so shranjeni na varnem mestgu(zaščiteni pred uničenjem)

Verodostojnost: od potrditve ti podatki niso bili spremenji.

Avtentičnost: Z gotovostjo lahko trdimo kdo je avtor.

Neovrgljivost: Avtor nemore zanikati avtorstva

Časovna opredeljenost: Z gotovostjo lahko ugotovimo kdaj je bila zagotovljena njegova verodostojnoist ter avtentičnost.

Kaj je šifriranje?

Šifriranje je postopek, ki razumljive podatke pretvori v oblikoe, ki je v splošnem nerazumljiva.

Razloži terminologijo šifriranja:

Čistopis – nešifrirani podatki

Šifropis – šifrirani podatki

Na čem temelji šifriranje sporočila?

Tajnost šifriranega sporočila mora temeljiti ma tajnosti ključa.

Kako lahko razdelimo šifre?

Glede na vrsto ključa:

Simetrične šifre

Asimetrične šifre

Glede na potek šifriranja:

Pretočne šifre – vsak video sprejem skupaj zakodiramo in pošljemo in ta postopek traja zelo dolgo(nekaj ur).

Blokovne šifre - šifrirajo se bloki in ne posamični znaki (zašifriramo dele dokumenta)

Glede na osnovni princip

Tajnost šifrirnega postopka

Tajnost ključa

Simetrični šifrirni algoritem: Le en ključ

Asimetrični algoritmi: dva ključa: en je javni drug pa zasebni

Kako lahko napademo neko šifro?

Tako da poznamo ena ali več šifropisov, znan šifropis in šifra…Dejstvo je da se npr. Nek pošiljatelj in prejemnik izmenjata podatke. Iz tega lahko dobimo delček ključa…ali pa delček nekega šifropisa. To je naš začetek na katerim gradimo naš napad.

Monoalfabetska šifra?

Naključno premešamo abecedo v ključu. Napad možen s frekvenčno analizo.

Polialfabetska šifra?

Uporaba več različnih monoalfabetskih ključev. Napad je možen z poglobljeno frekvenčno analizo.

Viegenerjeva šifra?

Uporaba velika števila monoalfabetskih ključev za šifriranje. Močna šifra. 300 let je veljala za nezlomljivo.

Razlika med Avtentikacijo pa Avtorizacijo?

Do kakšnih omrežnih napadov lahko pride?

Napad DOS, Zlonamerna programska oprema, spletno ribarjenje

Kako se lahko zaščitimo pred napadi iz omrežja?

Protivirusna programska oprema, Firewall je na ON, uporaba proti vohunske programske opreme, posodabljanje Sistema…

Opiši DES algoritem?

To je Data Encryption standard. Je simetrični šifrirni algoritem. Načrtovan za digitalne podatke. Je hiter in učinkovit. Zahteva skupni ključ.

Kaj je kongruenca?

To je ekvivalečna relacija deljivosti v množici celih števil kjer se deljenci velikokrat obnašajo kot ostanki the števil pri deljenju z deliteljem.

Graphical user interface, text, application

Description automatically generated

Kako deluje RSA šifriranje?

Vzamemo dve veliki praštevili p in q.

Izračunamo n = p \* q pa m = (p-1) \* (q-1)

Naključno izberemo število e, tako da e in m nimata skupnih deliteljev razen 1

Poišči število d med 0 in m tako da velja (e\*d) % m = 1

Javni ključ (n,e)

Zasebni klju (n,d)

ŠIFRIRANJE:

C = Pe % n

DEŠIFRIRANJE:

M = Cd % n

Kaj razlika med SSL in TLS?

SSL ali Secure Sockets Layer je tehnologija, ki zavaruje povezavo za prenos podatkov od pošiljatelja do prejemnika. To je zelo uporabna zadeva, saj moramo vedno zaščiti geslo, kodo bančne kartice(ko pošiljamo čez internet). TLS pa je ubistvu nadgradnja SSL-a vendar vedno ko zakupimo del strežnikov in domeno se sklicujemo na izraz SSL saj je bolj uveljavljen.

* Uporabljata RSA in DES
  + deluje po principu odjemalec/strežnik
  + strežnik pošlje avtentikacijski certifikat in svoj javni ključ
  + odjemalec pošlje svoj DES ključ šifriran po RSA
  + nato se podatki pošiljajo preko (skupnega) DES ključa