

PODZIMNÍ SOUSTŘEDĚNÍ KSP 2018 – SEZNAM PŘEDNÁŠEK

Tento spisek jest nabídkou přednášek, které byste na soustředění mohli slyšet, čili jakási obdoba matfyzácké Karolínky (ta je ale, pravda, ještě stále o něco tlustší). Přednášek je daleko víc, než kolik se dá za pár dní stihnout, a tak je na vás, abyste si vybrali, o které máte opravdu zájem. Pokud byste rádi slyšeli ještě o něčem dalším, klidně si o to napište (např. na fórum), třeba se najde někdo, kdo by vám o tom rád pověděl. Berte a vychutnávejte!

Údaje o jedné přednášce vypadají asi takto:

Stručný úvod do základů teorie vlkodlaků (“*Za dne ukryt v hloubi lesa, děs temný zvečera se plazí. . .*”) **LYK**

RNDr. Á. Cula

Úvod do moderní teorie vlkodlaků, čili též praktická daemologie a naiadologie.

Předpoklady: Měsíc v úplňku.

Dozvíte se (čteno v obvyklém pořadí): jméno přednášky, v uvozovkách motto přednášky, kód (pro snadnější odkazování na konkrétní předměty), jméno přednášejícího a nakonec stručný obsah přednášky. Hvězdičky znamenají obtížnost.

Základní přednášky

V této kategorii sídlí přednášky, které se dají považovat za základní stavební kameny informatiky, ať teoretické, či praktické.

Algoritmy a datové struktury

Základní algoritmy a jejich složitost (“*Čím menší je časová složitost algoritmu, tím větší je složitost kódu.*”) **ZAKL**

Pravděpodobně dvoudílná přednáška pro ty, kdo potřebují dohnat základní znalosti nutné pro ostatní přednášky. Zdefinujeme si základní pojmy jako je algoritmus, program, rekurze a jak se počítá jejich časová složitost, bude následovat přehled základních algoritmů – převážně třídění, rychlé hledání k -tého nejmenšího prvku, práce s výrazy a další.

Grafy & algoritmy (“*Pojďme si hrát s obrázky*”) **GA**

Co to jsou grafy, jak je v programech reprezentovat a hlavně k čemu se dají použít. Prohledávání grafu do šířky i do hloubky. Hledání nejkratších cest: Dijkstrův a Floydův algoritmus. Minimální kostry a Union-Find problem.

Prohledávání do hloubky **DFS**

Trochu hlubší pohled na prohledávání do hloubky. Jeho (často dost nečekané) aplikace v dalších algoritmech, jako je třeba hledání mostů, topologické třídění, rozklad na komponenty silné souvislosti či kreslení grafu jedním tahem.

Těžké problémy * **HARD**

Ríša Hladík

V rámci této přednášky se budeme zabývat problémy tak těžkými, že nikdo na světě pro ně neumí vymyslet efektivní (rozuměj polynomiální) algoritmus. Spousta lidí dokonce věří, že to vůbec možné není. Abychom mezi tyto problémy pronikli, seznámíme se s pojmy NP-úplnosti a NP-těžkosti. Především si však konkrétní těžké úlohy ukážeme a naučíme se i některé těžké úlohy rozpoznat. Závěrem si řekneme, jak se s těžkými úlohami vypořádat v praxi.

Toky v sítích (“*Když je v grafu povodeň, těsní?*”) **TOKY**

Kuba Maroušek, Martin Mareš

K čemu je dobré, když grafem teče voda. Předvedeme si klasický problém toků v sítích a jeho všelijaké, mnohdy dosti překvapivé aplikace. Jak rozestavět n věží na šachovnici a jak ji místo toho pokrýt dominovými kostkami? Další souvislosti, jako třeba násobná souvislost grafů.

Předpoklady: Umět plavat (zejména v matematice)

Datové struktury pro začátečníky (“*Pole oraná a neoraná, stromy ovocné a okrasné.*”) **DS1**

Jak si ukládat data natolik šikovně, abychom je nejen neztratili, ale také našli dříve, než si pro nás přijde Smrt. Klasické struktury jako pole, seznamy, fronta a zásobník, trie, vyhledávací stromy (vyvážené, AVL, a - b , splay), haldy (binární a obecně regulární) a v neposlední řadě hešování.

Datové struktury pro pokročilé * (“*Haldy a jiné kupky.*”) **DS2**

Kuba Maroušek, Ríša Hladík, Martin Mareš

Důmyslnější varianty vyhledávacích stromů: splay stromy, BB - α stromy, rankové stromy, vícerozměrné stromy. Chytřejší haldy: binomiální, Fibonacciho, rank-pairing. Amortizovaná analýza složitosti. Též několik přátelských randomizovaných datových struktur: skip listy a treapy.

Intervalové stromy * (“*Já bych ty intervaly nejradši. . . dal do stromu!*”) **ITREE**

Ríša Hladík

Intervalový strom je datová struktura pracující s intervaly, se kterou se můžeme setkat v mnoha úlohách (zejména soutěžních). Řekneme si, co to intervalový strom je, jaké všechny druhy intervalových stromů existují a jejich použití si ukážeme na úlohách. Na závěr si představíme jednu „magickou“ datovou strukturu jménem Fenwickův strom.

- Dynamické programování** (*“Kampak jsem si to jenom schoval?”*) **DYNP**
Vojtěch Sejkora, Martin Mareš
 Dynamické programování je programátorská technika využívající velice prostinkého nápadu: Proč něco počítat několikrát, když to mohu spočítat jednou a výsledek si uložit? Na této přednášce si ukážeme, že tento jednoduchý nápad může pomoci efektivně vyřešit i poměrně obtížné úlohy.
- Hledání v textu** (*“»Vyšíváme v seníku!« – kde jsem to jen viděl?”*) **TEXT**
Kuba Maroušek, Jirka Setnička
 Někdy potřebujeme najít podřetězec ve velkém množství textu. Stromeček trochu připomínající ten biologický aneb trie. Proč se ve vstupu vracet neboli Knuthův-Morrisův-Prattův algoritmus. Hledání více řetězců najednou podle Aha a Corasickové. Okénkové hešování Rabina a Karpa.
- Parsing čili analýza textu** (*“ $1+2*4 = 12$ ”*) **PARSE**
Standa Lukeš, Martin Mareš
 Často potřebujeme načíst nějaký složitý textový vstup: matematický výraz, webovou stránku v HTML, zdroják programu, . . . Ukážeme si, jak texty analyzovat (neboli parsovat), aniž bychom v nich zabloudili: rozdělení na lexikální a syntaktickou vrstvu, železničářský algoritmus na parsování výrazů, popis syntaxe pomocí regulárních výrazů a gramatik.
- Geometrie a počítače** (*“Nerušte mé kruhy! (ani jiné kvadriky)”*) **GEOM**
Jirka Setnička, Martin Mareš
 Základní algoritmy pro řešení geometrických úloh – konvexní obal, dva nejbližší body v rovině, výpočet obsahu nekonvexního mnohoúhelníka, lokalizace bodu, scanline algoritmus a jeho použití, Voroného diagramy a souvislost s persistentními datovými strukturami.
- Amortizace** (*“Cepek bývá daleko menší než součet částí.”*) **AMORT**
Standa Lukeš, Martin Mareš
 Spousta algoritmů je mnohem rychlejší, než jak na první pohled vypadají. Šikovní způsob, jak takové chování zkoumat, je amortizovaná časová složitost. Předvedeme několik trochu překvapivých příkladů amortizace: dvojková a jiná počítadla, datové struktury založené na přebudovávání, vyhledávací stromy bez otravného vyvažování, dynamizace datových struktur, udržování historie.

Programovací jazyky a nástroje

- Programování v jazyce C** **C**
Vojtěch Sejkora, Kuba Maroušek, Ríša Hladík
 Jazyk C patří k nejrozšířenějším jazykům, hodí se pro low-level programování i kusy kódu, které mají zejména být rychlé. Představíme si datové typy a běžné programové konstrukce, vysvětlíme si základy práce s ukazateli a také se seznámíme se standardními knihovnamí jazyka C.
- Objektově orientované programování** (*“Object-oriented system. If we change it, users object.”*) **OOP**
Vojtěch Sejkora, Kuba Maroušek
 Objektově orientované programování přináší jiný náhled na návrh řešení problémů. Vysvětlíme, jak se liší objektové a procedurální programování. Co je to objekt a co třída. Základní vlastnosti objektů (dědičnost, zabalení, polymorfismus). Co je to metoda, překrývání metod, virtuální metody (pozdní vazba) a čisté virtuální (abstraktní) metody. Jak se liší OOP ve statických (C++, C#, Java) a dynamických (Python) jazycích. Jak programovat objektově i bez podpory jazyka, třeba v Céčku.
Předpoklady: Znalosti procedurálního programování, například v Pascalu, v Pythonu nebo v C.
- Programování v jazyce Java** **JAVA**
Kuba Maroušek
 Java je jeden z nejrozšířenějších objektových programovacích jazyků za posledních deset let. Na přednášce se seznámíme s jeho myšlenkou a naučíme základy. Přednáška je dělaná pro posluchače, kteří umí alespoň základy jiného programovacího jazyka.
- Programování v jazyce C#** (*“Co se stane, když strčíme Céčko za mřížku?”*) **CIS**
Vojtěch Sejkora, Kuba Maroušek
 C# je moderní objektově orientovaný jazyk, jehož tvůrci se inspirovali přednostmi a úskalími ostatních programovacích jazyků, zejména Javy. Je jednoduchý a crossplatformní (tedy snadno v něm vytvoříte i okýnka, která nepoběží jen na okýnkách). Naučíme se základy a možná si i napíšeme jednoduchý prográmeček.
Předpoklady: Tušit něco málo o objektovém programování.
- Python** (*“print("Ffff".decode("rot13"))”*) **PYTH**
Kuba Maroušek, Vašek Šraier
 Jak programovat v Pythonu a jak v něm „nepsat Céčko“. Syntaxe, datové typy, funkce, třídy, . . . Na co si dát pozor, v čem se Python liší od ostatních jazyků a proč je mezi nimi tak oblíbený.

Perl (*“Jak Pejsek a Kočička vymýšleli programovací jazyk”*)

PERL

Pali Rohár, Martin Mareš

Jednoho dne se Larry Wall rozhodl, že nasype do jednoho velkého kotle spousty programovacích jazyků a unixových utilit, za stálého míchání povaří, posléze přecedí, přikoření a implementuje. Tak vznikl Perl, jazyk původně určený hlavně na zpracování textu, ovšem jak se ukázalo, též šikovný na spoustu dalších věcí. Asociativní pole, libovolně složité datové struktury za pomoci referencí, balíčky a objekty zdarma a hlavně regulární výrazy zde a všude. Zkrátka jazyk, který lze jedinečně milovat nebo nenávidět, nic mezi tím. Co se Perl 5 přiučil od Perlu 6.

Procesy a vlákna * (*“Koupil jsem dalších 15 procesorů, proč je to stále stejně pomalé?”*)

THREAD

Standa Lukeš

Jak vypadá víceprocesorové či vícejádrové PCčko a co to znamená pro programátora. Procesy, vlákna a úskalí komunikace mezi nimi. Jak se snese n kohoutů na jednom smetišti? Synchronizační primitiva: mutexy, semaforey, podmínkové proměnné. Spinlocky, deadlocky a livelocky. Jde to i bez synchronizace: atomické operace, transakční paměť. Které jazyky nám pomáhají a které spíš škodí. Kdy je lepší vlákna použít, a kdy ne.

Předpoklady: Trochu představy o hardwaru

Logické programování (*“Mohu být svým vlastním dědečkem?”*)

LOGP

Pali Rohár

Což takhle projednou neříkat počítači, jak má věci počítat, ale jenom mu zadat podmínky, které má výsledek splňovat? Neprocedurální programování vychází přesně z této myšlenky. Podíváme se na programovací jazyk Prolog, který vychází z formální logiky. Zjistíme, které problémy se v něm neobyčejně zjednoduší a které naopak programování promění v noční můru. Pokud jsi milovník rekurze, budeš u této přednášky nejspíš skoro celou dobu spokojeně vrnět.

Haskell (*“V téhle proměnné je uložen okolní svět.”*)

HASK

Kuba Maroušek

Základní kurz Haskellu – moderního funkcionálního jazyka. Ukážeme si syntaxi, vysvětlíme typovou kontrolu a typový systém. Přičichneme k třídám, zrušíme výjimky a zavedeme zcela bezpečná vlákna. Řekneme si, proč v Haskellu nejde komunikovat s okolním světem a proč nám pomůže si okolní svět uložit do proměnné. A že vlastně v Haskellu žádné proměnné nejsou, jen visačky na datech.

Předpoklady: Sklony k algebraickému chápání vesmíru, odvahu tváří v tvář své vlastní tváři a rekurzi.

SQL databáze (*“SELECT something FROM knowledge LIMIT 90min”*)

SQL

Kuba Maroušek, Standa Lukeš, Martin Mareš

Jak si schovat data do relační databáze a jak je tam zase najít a ideálně se zeptat rovnou na to, co chci vědět. Definice tabulek a indexů. Dotazy a jejich skládání a vnořování. Pohledy, funkce a triggery. Rozdíly mezi dialekty SQL.

Hardware a operační systémy

Principy počítačů (*“A opravdu uvnitř počítače běhají malí trpaslíci?”*)

HW

Jirka Setnička, Standa Lukeš, Martin Mareš

Vydáme se do země skřítků, kteří pohánějí počítače. Počítačové architektury od hodinok po superpočítač od Craye, jejich křivolaká historie i současnost. Co je to procesor, jak se programuje a jak se chová. Různé druhy paměti a jejich cacheování. Jak procesory komunikují s okolím – sběrnice, čipové sady, vstupní a výstupní zařízení. A co když je procesorů několik, nebo třeba pár tisíc? Přednáška bude praktická: pár počítačů při ní rozebereme a možná i nějaký postavíme.

Od zdrojáku k programu (*“Před spuštěním program přeložte. Stačí třikrát podělně?”*)

KOMP

Pali Rohár, Standa Lukeš, Martin Mareš

Mezi programem, který jste právě dopsali, a tranzistory uvnitř vašeho procesoru leží obrovské území obývané překladači, linkery, knihovníky, operačními systémy, loadery a jinými bájnými bytostmi. Pojďme zjistit, co jsou zač a co všechno s programem provádějí. Co udělá kompilátor za nás a co musíme naopak udělat my za něj.

UNIX (*“UNIX gives you enough rope to hang yourself.”*)

UNIX

Kuba Maroušek, Jirka Setnička, Ríša Hladík

Unixové operační systémy (zejména Linux) dobývají svět. Jak fungují uvnitř a jaké nabízejí výhody? Unixová filosofie a historie. Proč je systém složený ze spousty malých a jednoduchých kousků stabilnější a bezpečnější? Proč ovládání prostřednictvím textových příkazů je často efektivnější než klikátka? Jaké to je mít svůj systém pod kontrolou a „vidět mu pod ruce“? V čem spočívá moc textových souborů?

Skriptování v shellu (*“man 1 woman ... man 2 woman ... man group”*)

SHELL

Kuba Maroušek

Praktičtější zaměřená přednáška než UNIX, zabývající se hlavně tím, jak efektivně používat příkazovou řádku. Ukážeme si na spoustě příkladů, jak nám může automatizace všedních činností ulehčit život a jak silné nástroje pro ni UNIXový shell (který navzdory svému názvu existuje i pro Windows) svou jednoduchostí a flexibilitou poskytuje. Budeme spojovat spoustu jednoduchých příkazů do mocných celků a s nimi plnit i na první pohled komplexní úlohy, jako třeba automatické stahování a parsování věcí z webu. Některé činnosti vyžadují lidskou nápaditost a vhled. Ty ostatní bychom měli přenechat strojům.

Programování v Linuxu (“Všechno na světě je tak trochu soubor”)

PLX

Pali Rohár, Martin Mareš

Jak vypadá rozhraní mezi jádrem Linux a uživatelskými programy. Co se doopravdy stane, pokud ve svém céčkovém programu zavoláme `printf` nebo `malloc`. Jak napsat program, který vůbec nepotřebuje standardní céčkovou knihovnu. Co všechno se umí chovat jako soubor a co jako signál.

Předpoklady: Schopnost přečíst a napsat jednoduchý program v C.

Sítě a bezpečnost

Sítě a Internet (“Sítě nejen na ryby.”)

NET

Kuba Maroušek, Jirka Setnička, Martin Mareš

Jak funguje Internet a počítačové sítě vůbec: od elektronů v drátech (fotonů v optických kabelech nebo elektromagnetických vln) přes `packets` a jejich `forwarding` až k jednotlivým síťovým službám. Adresace, `internetworking` a dynamický `routing`. Jak NAT zachránil i zničil Internet a proč se těšíme na IPv6.

Sítě II – protokoly a síťové útoky (“Jak si přečíst `maily`. . . `sousedovy maily`.”)

NET2

Kuba Maroušek, Jirka Setnička, Martin Mareš

Volné navázání na NET. Budeme si povídat o tom, co za data nám po síti běhá a jaké se k tomu používají protokoly – DNS, FTP, HTTP nebo třeba i mailové SMTP a IMAP. Zaměříme se více na ty nejpoužívanější (metody GET a POST v HTTP), nakousneme `cacheování` a `nadlábneme se cookies`. A pokud zbude čas, využijeme zranitelnosti některých protokolů a provedeme síťový útok.

Předpoklady: Základní povědomí o počítačových sítích

Webové stránky

WWW

Kuba Maroušek, Standa Lukeš

Co se děje za oponou, když do prohlížeče zadáte adresu svých oblíbených stránek? A jak si takovou stránku taky pořídít? Přelet nad protokolem HTTP, seznámení s HTML a předvedení kaskádových stylů. Jak fungují dynamické stránky od formulářů až po JavaScript běžící v prohlížeči.

Kryptografie (“*Gbgb arav zbp gnwan mcenin.*”)

CRYPT

Jirka Setnička, Standa Lukeš, Martin Mareš

Kryptografie čili tajuplná nauka o šifrách, jejich konstrukci a hlavně o jejich luštění. Přísně tajné. Šifrovací systémy jako `lego`: základními kostičkami nám budou `symetrické` a `asymetrické šifry` a `jednosměrné funkce`, stavět z nich budeme `kryptografické protokoly` na bezpečný přenos, `autentikaci`, `digitální podpisy` a třeba i na házení korunou po telefonu. Předvedeme `nerozluštitelnou šifru` a dokonce to o ní i dokážeme.

Teoretická informatika

Složitější složitost *

SLOZ2

Martin Mareš

Trochu hlouběji o složitosti. Přesná definice výpočetního modelu a velikosti vstupu. Složitost v nejlepším, nejhorším a průměrném případě; `amortizovaná analýza`. Jak dokázat, že úlohu nejde řešit rychleji, aneb `dolní odhady`. Porovnávání problémů pomocí `redukci`, problémy NP-úplné a ještě těžší.

Předpoklady: SLOZ

Evoluční algoritmy * (“*Já to dělat nebudu, ať to za mě udělají mravenci!*”)

EVA

Jirka Setnička

Evoluční algoritmy se inspirovaly strukturami chování v přírodě a na jejich základě pak (optimalizačně) hledají řešení těžkých problémů. Na přednášce určitě zazní `genetický algoritmus`, zmíníme jeho algoritmy a když zbyde čas tak si obecněji popovídáme o `algoritmech` pohybujících se ve velkých prostorech řešení.

Strojové učení * (“*Umí počítače přemýšlet?*”)

MACHINE

Kuba Maroušek

Úlohy, kterými se zabývá strojové učení. Co je to učení s učitelem a bez učitele? Ukážeme si základní postupy, jak počítač můžeme naučit rozpoznávat věci, které sami rozpoznat neumíme. Umí Facebook nebo Google poznat, co máme rádi, jakou máme náladu, nebo jestli jsme těhotní?

Modely počítačů (“*Nač Pentium? Máme Turingovy stroje!*”)

MODEL

Jirka Setnička, Martin Mareš

V HW se dozvíte, jak fungují „opravdové“ počítače, zde pro změnu na čem počítají teoretici. Všechny počítače jsou si rovny, jen některé jsou si rovnější. Turingův stroj obyčejný, `vícepáskový`, `nedeterministický` a `univerzální`. `Random Access Machine (RAM)` a `Pointer Machine`. Trocha `minimalismu` aneb stroj s `počítadly`. Až nám začne být smutno, pořídíme si klidně N^2 procesorů a `spřáhneme je do paralelního počítače (PRAM)`. Rychlé `paralelní slévání` a `třídění`. Pokud zbude čas, ukážeme si `buněčné` a `grafové automaty`, nebo třeba `dlaždičky` v koupelně.

Vojtěch Sejkora, Ríša Hladík, Martin Mareš

O jazycích přirozených, počítačových a matematických, jejich popisu a rozpoznávání. Začneme těmi nejjednoduššími: regulární jazyky a výrazy, konečné deterministické a nedeterministické automaty. Pak budeme stoupat po příčkách Chomského hierarchie, kam až to půjde. Jak výpočetně silný je třeba takový automat na kafe?

Matematické přednášky

Grafy bez algoritmů**GRAFY***Jirka Setnička, Ríša Hladík*

Teorie grafů trochu teoretičtěji. Různé druhy grafů a jejich vlastnosti. Stromy a lesy. Kreslení grafů jedním tahem. Princip sudosti a skóre grafu. Jaké speciální vlastnosti mají rovinné grafy a jak je lze obarvit šesti nebo možná i pěti barvami. Jak poznat, že dva grafy (ne)jsou isomorfní. Mosty, artikulace a ušaté lemma. Párování, střídavé cesty a Hallova věta.

Úvod do teorie čísel**NUT***Ríša Hladík, Martin Mareš*

Co a k čemu je teorie čísel. Počítání v kongruenci, Euklidův algoritmus a jeho použití. Konečná tělesa a Malá Fermatova věta. Prvočísla a Eratosthenovo síto. Čínská zbytková věta a její algoritmická verze. Jak si odvodit kritéria dělitelnosti.

Kombinatorika (*“Nemám rád faktoriály. Faktoriály nemám rád. Rád nemám faktoriály. . .”*)**KOMB***Martin Mareš*

Při navrhování algoritmů a počítání jejich složitosti narazíme na celou řádku zajímavých a ne úplně triviálních kombinatorických problémů, a tak se naučíme, jak na ně. Základní triky s faktoriály a kombinačními čísly, sčítání konečných a občas i nekonečných řad, rekurentní rovnice a princip inkluze a exkluze. Možná se také potkáme s Dlouhým, Širokým a poněkud zmatenou šatnářkou.

Jak vypadá zrcadlo v číslech**LIN1***Marek Černý*

Na přednášce se podíváme na to, jaká matematika stojí za grafickými transformacemi obrázků. Ukážeme si, jak můžeme otáčet předmět pomocí jeho zobrazení osovou souměrností, a budeme si hrát s dalšími transformacemi. Dozvíte se, jak vypadají matice a jak je násobit s vektory. Povíme si, co všechno stojí na lineární algebře a jaké problémy dokáže vyřešit.

Lineární algebra ***LIN2***Marek Černý*

Naučíme se hledat řešení soustav rovnic na papíře tak, abychom toho museli napsat co nejméně (Gaussova eliminace). Zabředneme hlouběji do vektorových prostorů a budeme hledat jejich hezké ortogonální a ortonormální báze.

Předpoklady: Je dobré vědět, co je to matice a jak se násobí (přednáška LIN1).

Rozšiřující přednášky

Mezi rozšiřujícími přednáškami se dají nalézt různé specifictější obory a zájmy, jakožto i těžší přednášky navazující na předchozí díly ze základních přednášek. Mezi nabízenými přednáškami si tak můžete vybrat obor svého zájmu a tomu se dále věnovat.

Algoritmy a datové struktury

Nejkratší a jiné cesty * (“*Všechny cesty vedou do Horní Dolní, jen některé přes Řím.*”)

CESTY

Jirka Setnička, Martin Mareš

O problému hledání cest v grafech trochu podrobněji. Obecné relaxační schéma, Bellmanův-Fordův a Dijkstrův algoritmus a jejich zrychlení pomocí různých datových struktur. Potenciálová redukce a heuristiky (třeba A^*), zaokrouhlování délek hran. Souvislosti s násobením matic: transitivní uzávěr, Seidelův algoritmus, Kleeneho algoritmus a regulární výrazy.

Datové struktury pro ještě pokročilejší ** (“*log log log log ... glo glo glo ...*”)

DS3

Jirka Setnička, Martin Mareš

Na přednášce si ukážeme některou z méně známých složitějších datových struktur. Pokud Ti ostatní přednášky přijdou moc jednoduché, tato je ta pravá pro Tebe.

Suffixové stromy ** (“*Jak obrátit řetězec naruby?*”)

SUFF

Martin Mareš

Suffixový strom je zajímavá datová struktura, pomocí níž jde vyřešit většinu řetězcových problémů v lineárním čase. Podíváme se, jak suffixový strom vypadá, k čemu se hodí a jak ho sestrojít. Též prozkoumáme několik příbuzných zvířátek, jako třeba suffixové pole a suffixový automat.

Stromové algoritmy * (“*Půjdeme na to od lesa*”)

TREES

Jirka Setnička, Martin Mareš, Ríša Hladík

Stromy jsou jednou z nejtypičtějších (a nejjednodušších) odrůd grafů. Ledacos pro ně umíme řešit mnohem rychleji než pro obecné grafy, tak se pojďme podívat, jak se to dělá. Předvedeme několik obecných technik pro práci se stromy: DFS očíslování, „vandalskou indukci“, intervalové reprezentace. Různé rozklady: heavy-light, Fredericksonův, separátorový a ST-stromy.

Top-stromy **

TOP

Jirka Setnička

Pokračování přednášky TREES. Ukážeme dynamickou reprezentaci grafů jménem Top stromy, pomocí které lze efektivně odpovídat na spoustu otázek o grafech, které se nám pod rukama mění.

Magické algoritmy * (“*Pokročilá magie není rozlišitelná od technologie.*”)

MAGIC

Martin Mareš

O algoritmech značně magických a nečekaných. Jak násobit n -ciferná čísla rychleji než v kvadratickém čase. Kouzlo na slévání setříděných posloupností v konstantním prostoru. Isomorfismus stromů pomocí přihrádkového třídění. Bitové kejklřství. Hledání největší díry.

Programovací jazyky

C for wizards * (“*1[x]++++x[1]*”)

CWIZ

Martin Mareš

Ponořme se do hlubin Cčka, snad až na samé dno. Typový systém: elementární typy, typové výrazy, automatické konverze a rozpad typů (pole vs. ukazatel). Pořadí vyhodnocování kontra pořadí side-efektů (priority, synchronizační body a volatile). Triky s preprocesorem. Návěští a příkaz switch. Všelijaké zrady (velikosti typů, zarovnání, $(a + b) + c \neq a + (b + c)$, ...). Dialekty Cčka od K&R až po nejnovější normu C11 a různá nestandardní rozšíření jazyka. Proč jsou objekty potřebnější v myslí programátorově než v jazyce a proč je C lepší než C++ 😊
Předpoklady: Pověštná znalost jazyka C.

Programování v assembleru

PASM

Martin Mareš

Jak programovat procesor přímo, aniž by vám do toho mluvily překladače, linkery a podobná verbež. Začneme obecně, ale soustředíme se hlavně na procesory rodiny x86. 32-bitová a 64-bitová instrukční sada, FPU a panoptikum vektorových instrukcí. Rozdíly mezi intelovskou a AT&T syntaxí. Jak spojit assembler s vyššími programovacími jazyky. Optimalizace kódu. Stručný úvod do systémových architektur IA32 a AMD64.

Jazyk Go

GOLANG

Jirka Setnička

Go je moderní kompilovaný jazyk (vyvinutý původně v Google), který se pokouší být takovým Cčkem na steroidech. Umí být podobně rychlý, zaručuje větší typovou bezpečnost, ale má i prvky dynamicky typovaných jazyků. Jeho velkou silou je velmi snadné provázání na existující kód v C/C++, systém balíčků distribuovaných převážně přes Github, funkce vracující libovolný počet hodnot nebo třeba vestavěná podpora Unicode a vestavěná hashovací tabulka. Také se silně dbá na coding-style pro zajištění snadné čitelnosti programů.

Perl 6 (“Slečno, mohu vám ukázat svou sbírku operátorů?”)

PERL6

Martin Mareš

Je to Perl, a přitom to Perl není. Co je to? Aneb jak to dopadne, když se pokusíme navrhnout programovací jazyk budoucnosti a inspirovat se přitom filosofií Perlu. Typový systém, pokud zrovna chcete. Objekty, třídy a metatřídy. Periodická soustava (meta)operátorů. Definování jazyka v sobě samém. A co se to stalo s regulárními výrazy? Jak vypadají implementace P6 a kdy je prozatím lepší programovat na papíře. Praktické cvičení ve stavbě vzdušných zámků a bydlení v nich.

Černá magie v C++ * (“Je dobré znát, co umí atomová bomba (a její datový typ), abychom ji nechtěli použít.”)

CPP

Marek Černý

Pokročilejší prvky C++ (šablony, přetěžování funkcí a operátorů, preprocesorové hacky, ...) a možnosti jejich (po|zne)užití. Jak s nimi vytvořit věci magické (smart pointers, lambda funkce), ale i na první pohled docela obyčejné, které si bohužel jinak pořídit nejde. Nahlédneme do vnitřností slavné knihovny Boost. Proč je třeba preprocesorem vygenerovat 512 šablonovaných typů jen, abychom mohli rozumně předávat pointery na metody? Jaká akrobacie je nutná pro obyčejnou statickou inicializaci seznamu? A mnohá další překvapení. Přednáška vás naučí psát v C++ složité věci a přesvědčí, že v něm nechcete psát ani jednoduché.

Předpoklady: základní znalost C++, staticky alokovaný kyblík

JavaScript

JS

Kuba Maroušek, Standa Lukeš

Každá chytrá webová aplikace musí dneska mít aspoň trochu (nebo co nejvíc?) JavaScriptu. Ukážeme si, jak v něm psát, rozhýbeme pomocí něj webovou stránku, předvedeme si, jak fungují funkce, objekty, moduly a tak. A taky jak to debugovat, jak si pořídit typovou kontrolu nebo novější featury než prohlížeče umí.

Předpoklady: WWW

Webové UI funkcionálně

REACT

Standa Lukeš

Vyrábět složitější aplikaci čistě v Javascriptu a HTML může být trochu peklo, tak si ukážeme, jak se v tom neutopit. Vyrobit si totiž magickou abstrakci a z každé části stránky uděláme jenom funkci, která dostane stav svojí části a vrátí HTML, které nám prohlížeč zobrazí. Pomocí toho se dá v aplikaci udělat hezky pořádek, tak si ukážeme jak se v tomto modelu dá naprogramovat všechno potřebné.

Předpoklady: JS

F#

FSHARP

Standa Lukeš

Něco mezi Haskellem a Perlem - striktní typový systém, primárně funkcionální, hromada syntaktických featur a dobrý na praseň build skriptů i složitější systémy. Ukážeme si zajímavé vlastnosti, nějaké funkcionální vychytávky nebo jak F# zkompileovat do Javascriptu (a že to není tak marný nápad).

Základy psaní pro Android

ANDROID

Jirka Setnička, Vašek Šraier

Android se v našich mobilech, tabletech, netboocích a ledničkách zabydlel, a asi se v nejbližší době neodstěhuje. Na nějakém jednoduchém zadání si ukážeme, jak se pro něj programuje. Vlastní notebook s nainstalovaným Android Studiem se bude hodit.

Předpoklady: Aspoň pasivní znalost Javy.

Jazyk Rust

RUST

Standa Lukeš & Vašek Šraier

Rust je moderní programovací jazyk zaměřený na typovou a paměťovou bezpečnost, nekompromisní výkon, multithreading bez pastí. Zároveň přejímá nějaké vlastnosti funkcionálních jazyků se zachováním výborného výkonu, například iterátory jsou stejně rychlé jako for cykly. Ukážeme si základní principy jazyka. Jak se spravují objekty bez garbage collectoru a co nám to umožňuje (a co zakazuje).

Předpoklady: Schopnost programovat, tušení o nízkourovňových paměťových věcech.

Programovací nástroje a techniky

Git a jiné systémy pro správu verzí (“U svatýho tučňáka, kdo sem napsal tohle? Ono to tvrdí, že JÁ?!”)

GIT

Kuba Maroušek, Jirka Setnička, Pali Rohár

Jak vyvíjet program delší dobu a nezbláznit se u toho. Různé systémy pro správu verzí od diff/patch přes CVS a SVN až ke Gitu. Jak Git funguje: stromy, commity, větve, tagy. Merge mezi větvemi nebo mezi různými počítači.

Git pro pokročilé (“In case of fire, commit, push, and exit the building.”)

GIT2

Jirka Setnička, Pali Rohár, Martin Mareš

Používáte Git pro všechny své programy a k svačině místo novin čtete commit logy svých oblíbených projektů? V tom případě pojďme nahlédnout pod pokličku, jak Git funguje uvnitř. Reprezentace historie pomocí hešování grafů. Pracovní strom, index, commity a jejich adresy, větve. Pack files jako elegantní způsob komprese dat na disku i na síti. Kouzelnické triky: hledáme bugy půlením historie, přepisujeme dějiny, automaticky konvertujeme soubory. Git v praxi: jak se liší správa zdrojů v projektech o jednom, deseti a tisíci programátorech. Udržujeme patche k cizímu programu aneb StGit.

Gdb a jiné ladící nástroje * (“*Jak se ladí kytara, jak křišťálová koule a jak program (řazeno dle obtížnosti)*”) **GDB**

Pali Rohár, Martin Mareš

Kdo píše programy, které vždy hned fungují, ať se přihlásí. A kdo ne, ať se přihlásí na tuto přednášku. Ukážeme si několik nástrojů, jak si pomoci z nejhoršího. Mezi nimi třeba gdb, řádkový debugger (odvšivovač), strace, nebo valgrind. Kdy je použit a kdy se více hodí printf. Proč assert je tak užitečná věc.

Textový editor Vim (“*Víš, jaký je nejlepší textový editor? Vim.*”) **VIM**

Martin Mareš

Odložme na chvíli své myši a pojďme si vyzkoušet textový editor, který umí poslouchat na slovo. Pravda, budeme se ta slova muset chvíli učit, ale výsledek bude proklatě efektivní. Základní příkazy, práce s regulárními výrazy, makra, kouzla. Vimovité ovládání jiných programů, třeba webového prohlížeče.

Jak se nestat vepřem (“*/* You are not expected to understand this */*”) **STYLE**

Kuba Maroušek, Standa Lukeš, Martin Mareš

Tvrdí se, že čistý kód je mnohdy těžší, než ho psát – dokonce i po sobě, stačí krátká doba. Je několik obecně uznávaných pravidel, jak kód psát a jak ne, aby byl hezký a dobře čitelný. Od základních (rozumná pojmenovací konvence, systematické odsazování), až po to, kdy opravdu použít goto, jak členit program na funkce a jak využít nějaké třídy, moduly a podobně. Jak napsat užitečný komentář nebo dokumentaci. A kdy se vyplatí se na všechna tato pravidla vybodnout.

Testování a kvalita softwaru **QA**

Standa Lukeš

Výroba software není zdaleka jenom o programování. Ukážeme si, jak psát kód tak, aby nejenom fungoval, ale aby vyzařoval krásu a pokoj všem programátorům dobré vůle. Povíme si o různých způsobech testování, o tom jak udržet v kódu pořádek a dalších nástrojích, které pomáhají vyvíjet kvalitní software.

Předpoklady: Znat aspoň jeden příčetný programovací jazyk

Funkcionální (immutable) datové struktury **IMMUT**

Standa Lukeš

Datové struktury které se nikdy nemění ale lze efektivně vyrobit novou, která bude nějak změněná. K čemu bychom něco takového chtěli? Proč to asi nebudeme moc často potřebovat na vývoj algoritmů, ale může se náramně hodit na efektivní uchování dat při běžném programování. Jo, a taky si ukážeme jak se dají efektivně naimplementovat.

Hardware a operační systémy

Cache-oblivious algoritmy (“*Kešuješ, kešuje, kešujeme*”) **CACHE**

Jirka Setnička, Martin Mareš

Dnešní procesory mají několik úrovní vyrovnávacích pamětí (cache), což způsobuje, že ačkoliv si jsou všechny části paměti rovny, některé si jsou rovnější. Jak taková cache funguje? Jak se procesor rozhodne, co si v ní zapamatuje a co vyhodí? Jak toho můžeme využívat při programování, aby naše programy běžely rychleji? Předvedeme kousek teorie i několik praktických ukázek s poněkud překvapivým chováním.

Předpoklady: Kešu oříšky

File systémy (“*Opravdu je FAT tabulka tlustá?*”) **FS**

Kuba Maroušek, Pali Rohár, Martin Mareš

Povídání o tom, co leží mezi nulami a jedničkami na plotnách disku a přátelskou adresářovou strukturou našeho OS. Jak funguje FAT a jeho varianty (VFAT, FAT32). Tradiční Linuxové file systémy od EXT2 k EXT4. Nadějný nový BtrFS, který je možná za pár let nahradí. Co se hodí na SSD.

Digitální elektronika aneb jak se dá z hradel vyrobit procesor **DIGI**

Standa Lukeš, Jenda Hadrava

Jak fungují digitální elektronické obvody, ze kterých jsou postavené (nejen) počítače. Postavíme si nějaké kombinační obvody - transistory, hradla, multiplexery, sčítačky. Pak nějaké registry, které budou držet stav. A nakonec to pospojujeme dohromady jak se pomocí tohoto dají interpretovat instrukce.

Jak postavit (rychlý) procesor **CPU**

Standa Lukeš

Vysypeme z rukávu jak se dá vyrobit procesor a půjdeme ho zrychlit. Proč nejde jednoduše zvýšit frekvenci, a jak to teda udělat. Jak ušetřit čas při přístupu do paměti, na co je ta slavná predikce skoků a jak se dá spouštět víc instrukcí najednou.

Předpoklady: DIGI

Paralelně bez zámků **CAS**

Standa Lukeš

Jak vyrobit datovou strukturu která je v konzistentním stavu vždycky, když se na ni člověk podívá. Začneme s nějakými zřejmými věcmi – přičítáním čísla, výměnou jedné globální proměnné a tak. Pak si ukážeme skip list – něco jako strom, akorát jednodušší a paralelní. Nakonec se můžeme podívat na nějaké složitější věci, třeba hashovací tabulky nebo B-stromy.

Vašek Šraier

Podíváme se do hlubin Androidu. Do míst, kam se většina lidí nedostane. Projdeme si celý bootovací proces, zamyslíme se nad zabezpečením systému. Vysvětlíme si, co obnáší rootnout si systém a jaké problémy při tom mohou nastat. Podíváme se na to, co se děje při běhu aplikací za normálního stavu. Bude se hodit notebook a zařízení s Androidem.

Předpoklady: Tušení o základním fungování Linuxových systémů, orientace v shellu.

Sítě a bezpečnost

E-mail (“Drahoušek zákazník.”)

EMAIL

Pali Rohár

Co se stane s e-mailem, když jej odešlete? Kudy chodí a kudy jej čerti nesou? Jaké máte záruky, že přijde; proč občas přijde pozdě nebo vůbec. Problém formátů a kódování, chyby webových i jiných klientů. Protokoly SMTP, POP, IMAP a co se stane, když do nich přimícháme SSL/TLS. E-mailová bezpečnost, SPAM a (nefunkční?) obrana pomocí SPF, DKIM a DMARC. Nakonec se podíváme na ne zrovna triviální grafový problém, který je v emailech skrytý.

IPv6 (“Viac adries znamená aj viac problémov”)

IPV6

Pali Rohár

Internetoví provideri nám pomalu začínají ponoukat připojení do světa IPv6 Internetu. Čo to IPv6 je? Čím sa líši od svojho staršieho kolegu IPv4? Prečo by sa o neho mal zaujímať aj obyčajný užívateľ? Pozrieme sa aké problémy IPv6 rieši, aké má výhody oproti IPv4 ako aj na súčasný stav podpory IPv6 v operačných systémoch. A hlavne s akými problémami sa človek bežne stretne než si bude môcť doma spraviť IPv6 sieť.

Předpoklady: NET

Bezpečné programovanie (“V každom programe je aspoň jedna bezpečnostná chyba, tak poďme to napraviť”)

SECPRG

Pali Rohár

Dosť často sa programy nepíšu iba na jedno spustenie, ale sa používajú dlhé roky. Veľa z nich navyše bude používať úplne niekto iný. Na tejto prednáške sa pozrieme na časté chyby v programoch a ako im predchádzať. Ako klasifikovať či chyba je závažná a bezpečnostná. Ďalej si povieme, čo robiť v prípade, ak nájdeme nejakú (možno bezpečnostnú?) chybu. Komu ju oznámiť a komu radšej nie.

Aplikace kryptografie * (“6140 a184 c9a6 41f1 de99 e733 354a f451”)

CRYPT2

Martin Mareš

Pokročilejší a občas nečekané aplikace základních kryptografických primitiv. Jak přesvědčit server, že známe heslo, aniž bychom mu ho posílali? Jak zajistit, aby útočník nemohl dešifrovat komunikaci, ani když dodatečně získá soukromý klíč? Jak funguje BitCoin (decentralizovaná digitální měna) či Tor (protokol znemožňující komunikaci po cestě vědět, kdo s kým komunikuje)?

Předpoklady: Základní povědomí o šifrování (CRYPT) a víra v existenci náhodných čísel

Praktická kryptografie (“A proč jsou všechny ty zámky na papírových dveřích?”)

PCRYPT

Martin Mareš

Programátoři si často myslí, že pro bezpečnou komunikaci stačí vybrat si z knihovny osvědčenou silnou šifru. Jak naivní! Navrhnout bezpečný protokol není maličkost a dá se při tom ledacos zpackat. Replay útoky (jak otevřít auto krabičkou za 30 dolarů), útoky na padding a na blokovou strukturu. Či že je ten podpis? Jak nepoužívat RSA a jak nehešovat hesla. Jak náhodná jsou vaše čísla? Postranní kanály: časování, spotřeba, záření. K čemu se crackerům hodí termoska s tekutým dusíkem.

Eliptické krivky a kryptografia

ECC

Pali Rohár

Kryptografia založená na eliptických krivkách sa vo veľkom nasadzuje a rozširuje po svete od smart kariet až po HTTPS servery. Na tejto prednáške si ukážeme, čo je sú to eliptické krivky, ako vyzerajú, ako sa s nimi dá počítat a ako ich použiť v kryptografii na šifrovanie alebo podpisovanie správ. Zároveň sa zoznámime s algebraickými pojmami ako sú abelovská grupa, či konečné teleso. Úvodná prednáška určená pre tých, ktorí ešte o eliptických krivkách nič nevedia.

Web uvnitř (“Error 402: Payment Required. Please insert a coin.”)

HTTP

Martin Mareš

Většina webu je dnes založena na protokolu HTTP, pojďme se podívat, jak funguje uvnitř. Metody GET, POST, ale třeba i PUT. Dohadování o typu dat. Cacheování, revalidace a transformace dat. Křupavé sušenky. Jak se vypořádat s dynamicky generovaným obsahem aneb protokoly CGI, WSGI apod. Mezi klientem a serverem aneb DNS a virtuální servery. Nakonec do toho všeho přimícháme SSL/TLS a máme HTTPS. Malá ochutnávka HTTP/2.0.

Grafika a typografie

Počítačová grafika (“Namaluj mi beránka. . .”)

GFX

Vojtěch Sejkora

Kreslení a zpracování obrazu na počítači. Co vše obnáší vykreslení obyčejné čáry, aby to bylo rychlé a pěkně vypadalo. A co teprve, když ta čára zatáhne! Vyplňování n -úhelníků a křivkou ohraničených oblastí, flood fill. Také maticové filtry pro zpracování fotek (zaostření, rozmazání), anti-aliasing a dithering. Pokud se stihne, tak navíc základy 3D vykreslování.

Barevné systémy (“*Co je na konci duhy?*”)

COLOR

Martin Mareš

O podstatě světla a barevného vidění a různých pokusech o reprezentaci barev v počítačích, fotoaparátech, televizích a podobných zařízeních. Systémy RGB, CMY(K), HSV, XYZ, Lab s jejich výhodami i neduhy. „Systém“ Pantone. Reálné kontra imaginární barvy aneb proč nejde vyfotit duha.

MetaFont, MetaPost (“*Teď ten obrázek takhle zkroučím a pak ho přeložím.*”)

MF

Jirka Setnička

Lehké nakousnutí jazyka, ve kterém můžete opravdu kreslit planimetrické obrázky, ale i třeba písma nebo piktogramy do zadání a řešení KSP. Jak vypadají CM fonty (ty, které používá \TeX) a jak se autorovi povedlo, že se z jediného „obrázku“ dá vygenerovat tlusté, tenké, rovné, skloněné, šišaté písmenko.

Typografie (“*What You See Is all What You’ve Got!?*”)

TYPO

Martin Mareš

Jak na počítači text nejen napsat, ale také vysázet tak, aby pěkně vypadal a aby (což je důležitější) se i příjemně četl. Jak se sází pohádka, jak báseň a jak vzorové řešení KSP plné komplikovaných vzorců. Jak jde dohromady staleté umění typografické a moderní technika. Přineste knihy i letáky, zkritizujeme sazeče, co se do nich vejde.

\TeX (“*No pages of output. Ask a \TeX nician.*”)

TEX

Jirka Setnička, Ríša Hladík, Martin Mareš

Z předchozí přednášky máme představu o tom, jak vypadá pěkná sazba. K její výrobě nám pomůže typografický systém \TeX . Praktická přednáška s ukázkami použití \TeX u od hladké sazby knihy až po zběsilosti hraničící s programováním. Jak do \TeX u vkládat obrázky a jak to raději nedělat. Kde shánět další informace: \TeX book, \TeX book naruby a další zajímavá literatura. Praktické rozdíly mezi různými dialekty \TeX u. Všelijaká rozšíření: pdf \TeX , e \TeX , Lua \TeX .

Formát PDF

PDF

Martin Mareš

Jeden z nejrozšířenějších formátů na předávání dokumentů má za sebou spletitou historii i dokumentaci. Ukážeme si, jak vypadá uvnitř a co se do něj dá uložit: grafické objekty, text, fonty, odkazy, všelijaké anotace a meta-data, a dokonce i kryptografické podpisy. Zmíníme se o profilech, třeba PDF/X a PDF/A. Při troše štěstí si vytvoříme jednoduchý PDF soubor ručně a možná půjde i otevřít.

Teoretická informatika

Persistentní datové struktury * (“*Datové struktury cestující časem*”)

PERS

Martin Mareš

Ukážeme si (téměř) obecný způsob, jak naučit datové struktury zapamatovat si celou svou historii. Předvedeme si, jak tuto historii modifikovat a k čemu to je celé dobré.

Třídy složitosti ** (“*Kolik sekund stojí jeden bajt?*”)

SLOZ3

Martin Mareš

Složitost opravdu důkladně: nejrůznější třídy složitosti a vztahy mezi nimi. Vztahy mezi časem a prostorem, odstraňování nedeterminismu a Savitchova věta. Jak víme, že všechny třídy nejsou stejné: dolní odhady a věty o hierarchii. Stroje s kvantifikátory, třída PSPACE a polynomiální hierarchie. Pravděpodobnostní třídy složitosti. Orákula a neuniformní složitost.

Předpoklady: SLOZ2

Pravděpodobnostní algoritmy (“*Kudy dál? Hoďme si kostkou!*”)

PPALG

Martin Mareš

Když nevíme, jak se v algoritmu rozhodnout, někdy pomůže ponechat to náhodě a prostě si „hodit kostkou“. Dokážeme sestavit algoritmy, které jsou rychlé, i když správný výsledek vydadí jen v 99 % případů. Ale i takové, které odpoví správně vždycky, ale rychlé jsou jen v průměru (třeba QuickSort). Těž ukážeme, jak pomocí náhody zabránovat kolizím v hešování.

Vyčíslitelnost ** (“*S Halting problémem na věčné časy!*”)

VYCIS

Martin Mareš

Některé problémy se dají vyřešit snadno, jiné obtížněji a některé dokonce vůbec. Obecněji: Ať si vymyslíte jakýkoliv rozumný programovací jazyk, vždycky existuje problém, který se v něm nedá vyřešit. Jak se ale dokazuje, že něco nejde? Matematický pohled na výpočetní modely a univerzální stroje, rekurzivně spočetné a rekurzivní množiny a funkce. Halting problem a diagonální důkazy.

Aplikace informatiky

Čárové kódy (“*Jak naučit počítače číst láhve od Coly*”)

BAR

Martin Mareš

Čárové kódy dnes potkáváme na každém kroku, ale jak doopravdy fungují? Prozkoumáme klasické jednorozměrné kódy (UPC, EAN, Code39, Code128), jakož i novější dvojrozměrné (QR, Aztec, DataMatrix). Kódovací a dekodovací algoritmy plus trocha matematiky okolo zabezpečení proti chybám. Další počítačem čitelné značky: RFID, bílé křížky na asfaltu, ...

Kompresie dat (“*Jnm idln kpln j nstlčtln.*”)

ZIP

Jirka Setnička, Martin Mareš

Pokud jsou data příliš velká, můžeme je zkusit zkomprimovat. Předvedeme základní kompresní algoritmy: triviální (RLE), slovníkové (LZ77), statistické (Huffmanovo a aritmetické kódování) a některé pokročilejší techniky, jako třeba Burrowsovu-Wheelerovu transformaci (BZIP). Zmíníme se o kompresi zvuku, obrazu a videa (prediktory, wavelety, všelijaká ztrátová komprese).

Matematické přednášky

Pravděpodobnost

PAST

Marek Černý

Jak pracovat s pravděpodobností matematicky. Ukážeme si pravděpodobnosti jevů, nezávislé jevy střední hodnotu, náhodné proměnné a další. Také si vše procvičíme na několika příkladech. Pokud zbyde čas, tak si také ukážeme, jak se dá pravděpodobnost využít v informatice.

Teorie množin a matematika nekonečen * (“*Kdo je nejvyšším z kardinálů?*”)

TEMNO

Martin Mareš

Historie matematiky je dlážděna trampoty s nekonečnem. Začalo to roztomilým problémem s želvou pana Zénona a vedlo až k poněkud děsivým paradoxům 18. století. V moderní době jsme se proti tomu obrnili teorií množin, na níž je dnes takřka celá matematika postavena. Jak se taková teorie buduje a jak se pomocí ní popisují nekonečné objekty. Množiny a jejich velikosti. Cantorův diagonální trik. Ordinály a houšť kardinálů. Potenciální kontra aktuální nekonečno. Jak si pořídít přirozená čísla a jak ta reálná. Potíže s axiomem výběru.

Teorie čísel a RSA * (“ $2^{67} - 1 = 193\,707\,721 \cdot 761\,838\,257\,287$ ”)

NUT2

Martin Mareš

Pokračování teorie čísel, které nás dovede až k RSA – asi nepoužívanějšímu asymetrickému šifrovacímu algoritmu dnešní doby. Počítání modulo složené číslo a Eulerova věta. Jak RSA funguje, proč funguje a jestli bude ještě fungovat. Generování klíčů, faktorizace kontra testování prvočíselnosti. Časová složitost aritmetiky.

Prvočíselné věty *

NUT3

Martin Mareš

Věty o rozložení prvočísel jsou tradičně považovány za jednu z nejmysterióznějších oblastí teorie čísel. Zde ukážeme, jak některé z nich odvodit snadným pozorováním vlastností kombinačních čísel: rozbor časové složitosti Eratosthenova síta, Bertrandův postulát („Mezi n a $2n$ je aspoň jedno prvočíslo.“), hustota prvočísel.

Fourierova transformace **

FFT

Martin Mareš

Jak rychle umíte násobit n -ciferná čísla? My to umíme lineárně. Hodí se k tomu chytrý trik pana Fouriera, který už dávno patří k matematické a fyzikální klasice. Ukážeme, co je Fourierova transformace zač, jak ji rychle spočítat a k čemu je dobrá: rychlé násobení polynomů i čísel, digitální zpracování zvuku a obrazu (spektrální analýza či třeba komprese).

Předpoklady: Základy komplexních čísel

Meta-matematika * (“*Tato věta sem nepatří.*”)

METAM

Martin Mareš

Pokud budeme v životě věřit všemu, co je „přeci zřejmé“, dostaneme se brzy do potíží a v matematice to platí dvojnásob. Přírodní vědy si vymyslely opakovatelné pokusy a matematici axiomatický přístup. Ukážeme, jak z jednoduché sady axiomů vybudovat takřka celou matematiku. Dokonce tak, že správnost důkazů za nás ověří počítač, aspoň když mu trochu pomůžeme. Nadšení trochu ochladí Gödelova věta: ať děláme, co děláme, vždy zbude nějaké nerozhodnutelné tvrzení. Pomůže přidávat axiomy? Asi ne, ale za odměnu získáme mnoho různých matematik. A dá-li bůh, stihneme dokázat jeho existenci i neexistenci ☺.

Teorie (vesměs samoopravných) kódů (“*f y cn rd ths, y wll b gd cmptr prgrmmr!*”)

KODY

Martin Mareš

Jak komunikovat po lince, která průměrně každý k -tý bit přeneše špatně? K tomu se hodí teorie samoopravných kódů, která nás naučí: vzdálenost slov a jejich souvislost s detekcí a opravou chyb, paritní a lineární kódy, perfektní kódy, Reed-Solomonovy a vůbec polynomiální kódy a několik dolních odhadů nádavkem. A jak s teorií kódů souvisí třeba čeština?

Komplexní a komplexnější čísla (“ $1 = \sqrt{1} = \sqrt{(-1)(-1)} = \sqrt{-1}\sqrt{-1} = i \cdot i = i^2 = -1$. Huh?”)

CPLX

Martin Mareš

Jak se nám matematika změní, když připustíme, že se záporná čísla také dají odmocňovat? Čísla imaginární a komplexní a jejich různé podoby. Součtové vzorce pro sin a cos dostaneme téměř zdarma. K čemu se hodí v matematice a k čemu ve fyzice. Proč se zastavit u dvou složek aneb kvaterniony, oktoniony a Cliffordovy algebry. Remember, life is complex.

Teorie nemožného * (*“Neexistence důkazu není důkazem neexistence. Dokažte.”*)

NONEX

Martin Mareš

Existenci slona v Africe snadno dokážete tím, že ho přivedete. Jak ale ukázat, že tam žádný slon není, případně že sice je, jenže ho nejde najít pomocí pravítka, kružítko a jeepu? Přímo se to dělá těžko, ale existuje spousta krásných triků, jak neřešitelnost problémů dokazovat. Nesložitelné hlavolamy, nerozvázatelné uzly, nepopsatelná čísla, neroztřetitelné úhly, nealgoritmické problémy a jiné slasti nekonstruktivní matematiky. Jak naopak ukázat, že něco existuje, aniž bychom věděli, jak to vypadá?

Úvod do matematické analýzy * (*“ $\lim_{8 \rightarrow 9} \sqrt{8} = 3$ ”*)

MA

Kuba Maroušek, Martin Mareš

Jak zjistit, jaký tvar má graf nějaké funkce? Jak najít její minimum? Jak spočítat délku spirály nebo objem sudu (třeba i čtyřrozměrného)? Jak spočítat $\sin x$ nebo třeba π ? Na to všechno se hodí limity, derivace a integrály. Nejprve si o nich vybudujeme jednoduchou geometrickou představu, pak je nadefinujeme pořádně a naučíme se s nimi počítat.

Catalanova a Fibonacciho čísla * (*“1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, ?”*)

CAT

Martin Mareš

Kolik existuje binárních stromů? Kolika způsoby jde uzavřít výraz? A kolika způsoby projít čtvercovou mřížku, aniž bychom překročili úhlopříčku? Kam oko pohlédne, všude se skrývají Catalanova čísla. Kromě případů, kdy za ně zaskakují čísla Fibonacciho. Povídání o dvou zajímavých posloupnostech a jejich početném příbuzenstvu. Dlouhá cesta od hezkého vzorečku k rychlému algoritmu.

Teorie sítí * (*“Co mají společného lidské vztahy a třídy v Javě?”*)

NA

Kuba Maroušek

Všichni máme rádi grafy, protože se dají dobře kreslit – až do chvíle, než nám na stole přistane graf s miliony vrcholů a ještě více hranami. Ukážeme si techniky, jak se v takových datech vyznat, jak si lépe představit jejich strukturu a co zajímavého z nich dokážeme určit. Je pravda, že s každým člověkem na zemi se znáte přes řetěz šesti různých lidí? Jak určuje Google důležitost webových stránek? A proč máme neustálý dojem, že naši kamarádi mají více kamarádů, než my sami?

Hausdorffův zvěřinec ** (*“Jaký objem má π -rozměrná koule?”*)

HAUS

Martin Mareš

Možná vás už také zarazilo, že některé fraktály nejsou ani dvourozměrné, ani třírozměrné, ale něco mezi tím. Pojdme se podívat, co to znamená. Cestou potkáme různé zajímavé partie matematiky (jako třeba metrické prostory a teorii míry) a různá podivuhodná zvířátka: Cantorovo diskontinuum, von Kochovu vločku a Hilbertovu křivku.

Půlnoční přednášky

Aneb přednášky přednášené (nejen) o půlnoci na různá zajímavá témata nejen o informatice. Pokud nějaká z nich nebude oficiálně vypsaná, je možné si konkrétního organizátora ve volné chvíli chytit a přesvědčit ho k přednášení.

Hippo:Web (*“Znovu vynalézáme kolo, ať konečně není hranaté.”*)

HIPPO

Kuba Maroušek, Martin Mareš

Organizátoři KSPčka rádi dělají pokusy na sobě samých, a tak při vývoji nového webu semináře pohrdli všemi vyzkoušenými webovými frameworky a napsali si potvrzku jménem Hippo. Trochu netradiční, ale v mnoha ohledech velice příjemnou. Namátkově: HTML coby datová struktura, triky s kryptografickými tokeny, kombinování programovacích jazyků a tvorba dynamických stránek, které jsou přitom skvěle kešovatelné. Případně také nahlédnutí do dalšího zákulisí KSPčka.

Organizování (*“Ten dělá to a ten zas tohle aneb co obnáší organizátorem být”*)

ORG

Jirka Setnička

Volné povídání o tom, co se všechno skrývá za organizováním různých seminářů a podobných akcí, primárně pak KSPčka. Jaká práce, jaké radosti a jaké starosti s sebou organizování nese, co se přitom člověk může naučit a také pár cenných rad do života. Jak se z toho nezbláznit a pár bláznivých příhod k tomu.

Základy první pomoci (*“Jak někomu zachránit život a jak málo k tomu stačí”*)

ZDRAV

Jirka Setnička

Pobavíme se o základech první pomoci. Jak správně vyhodnotit situaci a kdy je potřeba volat pomoc? Jak se postarat o člověka v bezvědomí, jak kontrolovat životní funkce a jak člověka stabilizovat do příjezdu pomoci? Ukážeme si, jak málo stačí k záchraně života a naučíme se nebát se první pomoci. A také, že naše bezpečí je v každé situaci na prvním místě.

Technika ve sci-fi (*“Scotty, beam me up”*)

SCIFI

Jirka Setnička

Technické objevy v různých sci-fi (Star Trek/Gate/Wars i dalších) a pohled na ně z perspektivy dnešních fyzikálních znalostí. Proč se v Hollywoodských filmech ozývá ve vesmíru zvuk laserů, i když je tam vakuum, a je možné cestovat rychleji než světlo? Také možná zabrousíme do některých filozofických otázek – primární směrnice o nevměšování ve Star Treku a jiné.

Létání a natáčení s dronem (*“Ne ne, to není řízená raketa, tam jen visí velký foťák.”*)

DRON

Jirka Setnička

Na co je dobré mít drona, kdy se dá reálně použít a kdy je to jen drahá hračka. Přehled o technologiích, jejich možnosti a omezení. Jak se liší armádní drony od dronů pro kameramany a nebo od dronů, které pořídíte za pár šupů v hračkářství. Také lehký výlet do legislativy – s čím smíme létat, kde smíme létat a co s tím můžeme dělat. Pokud počasí dá, tak i praktické hrátky s dronem.

Lingvistika (*“Přísudek je v této větě podmět.”*)

LING

Martin Mareš

Převážně nevážné a mírně nepřed-vídatelné po-vídání o jazyku i jazyce. Základní jazykové rodiny a jejich podobnosti i odlišnosti. Co má společného čínština s angličtinou a co nikoliv. Proč jeden jazyk potřebuje 15 pádů, zatímco jiný se bez nich obejde úplně. Jak se jazyky vyvíjejí a jak se navzájem ovlivňují. Kde se berou jazyková pravidla. Kde se vzalo písmo a proč se mluvený a psaný jazyk tolik liší. Jak se na jazyk dívá matematik a jak se na matematiku dívají lingvisté.

Fonetika (*“Pojďte, zachrochtáme si spolu!”*)

FON

Martin Mareš

Malá inventura zvuků, které lidé dovedou vytvářet, a jejich použití v komunikaci. Různé způsoby vytváření a modulace zvuku. Kolik různých B dokážete říci? Fonetické kontrasty a co si z nich různé jazyky vybraly. Rázy, polosamohlásky a jiní obyvatelé polosvěta. Přízvuk kontra délka. Asimilace, přehlasování a další „principy líné huby.“ Vše prakticky procvičíme.

Orientace

ORI

Martin Mareš

Jak ze neztratit v terénu a jak se neztratit na moři. Vývoj umění navigace. K čemu je důležité slunce a hvězdy, ale proč mořeplavcům nestačí, alespoň dokud neobjevíme hodinky. Použití mapy, busoly a GPSky. Orientace bez pomůcek a použití Ariadniny nitě. Bleskový úvod do sférické astronomie a časomíry čili jak (ne)postavit sluneční a třeba i měsíční hodiny. Jak reprezentovat mapu v počítači a jak raději ne. Jak zapisovat polohu místa na Zemi (přestože Země má tvar podivně nakousnuté hrušky) a kolika způsoby to jde. Různé druhy map a jejich (z)kreslení. Jak se neztratit v kartografii. Praktické cvičení v terénu.

Čaj (*“Jak vypadá odvar z nezralých pražců?”*)

TEA

Martin Mareš

Pojďme usednout k šálku lahodného čaje a povídat si o tom, co se v něm skrývá. Kde se čaj vzal, kde se pěstuje, jak se zpracovává a jak ho připravovat. Trocha čajového zeměpisu, dějepisu i čajové chemie a čajové kultury. Těž o všelijakých substancích čaji podobných.

Bezstátní společnost (*“Volný trh může zajistit vše, třebas i bezpečnost, nebo ne?”*)

ANCAP

Vojtěch Sejkora

Co to jsou daně. Co to je volný trh. Jaký je rozdíl mezi volným trhem a vlastněním firmy soukromníkem. A o tom, jak lze vybudovat společnost pouze na základě vlastnických práv a jejich neporušování.

Abecední seznam přednášek

LYK Stručný úvod do základů teorie vlkodlaků.. 1

Základní přednášky

AMORT	Amortizace	2	HW	Principy počítačů	3
DS2	Datové struktury pro pokročilé	1	THREAD	Procesy a vlákna	3
DS1	Datové struktury pro začátečníky	1	PLX	Programování v Linuxu	4
DYNP	Dynamické programování	2	JAVA	Programování v jazyce Java	2
EVA	Evoluční algoritmy	4	CIS	Programování v jazyce C#	2
GEOM	Geometrie a počítače	2	C	Programování v jazyce C	2
GA	Grafy & algoritmy	1	DFS	Prohledávání do hloubky	1
GRAFY	Grafy bez algoritmů	5	PYTH	Python	2
HASK	Haskell	3	SQL	SQL databáze	3
TEXT	Hledání v textu	2	SHELL	Skriptování v shellu	3
ITREE	Intervalové stromy	1	SLOZ2	Složitější složitost	4
LIN1	Jak vypadá zrcadlo v číslech	5	MACHINE	Strojové učení	4
AUTO	Jazyky, gramatiky a automaty	5	NET2	Sítě II – protokoly a síťové útoky	4
KOMB	Kombinatorika	5	NET	Sítě a Internet	4
CRYPT	Kryptografie	4	TOKY	Toky v sítích	1
LIN2	Lineární algebra	5	HARD	Těžké problémy	1
LOGP	Logické programování	3	UNIX	UNIX	3
MODEL	Modely počítačů	4	WWW	Webové stránky	4
OOP	Objektově orientované programování	2	ZAKL	Základní algoritmy a jejich složitost	1
KOMP	Od zdrojáku k programu	3	NUT	Úvod do teorie čísel	5
PARSE	Parsing čili analýza textu	2			
PERL	Perl	3			

Rozšiřující přednášky

CRYPT2	Aplikace kryptografie	9	MAGIC	Magické algoritmy	6
COLOR	Barevné systémy	10	METAM	Meta-matematika	11
SECPRG	Bezpečné programování	9	MF	MetaFont, MetaPost	10
CWIZ	C for wizards	6	CESTY	Nejkratší a jiné cesty	6
CACHE	Cache-oblivious algoritmy	8	CAS	Paralelně bez zámků	8
CAT	Catalanova a Fibonacciho čísla	12	PERL6	Perl 6	7
DS3	Datové struktury pro ještě pokročilejší	6	PERS	Persistentní datové struktury	10
DIGI	Digitální elektronika aneb jak se dá z hradel vyrobiť procesor	8	GFX	Počítačová grafika	9
EMAIL	E-mail	9	PCRYPT	Praktická kryptografie	9
ECC	Eliptické křivky a kryptografie	9	PAST	Pravděpodobnost	11
FSHARP	F#	7	PPALG	Pravděpodobnostní algoritmy	10
FS	Filesystemy	8	PASM	Programování v assembleru	6
PDF	Formát PDF	10	NUT3	Prvočíselné věty	11
FFT	Fourierova transformace	11	TREES	Stromové algoritmy	6
IMMUT	Funkcionální (immutable) datové struktury	8	SUFF	Suffixové stromy	6
GDB	Gdb a jiné ladící nástroje	8	KODY	Teorie (vesměs samoopravných) kódů	11
GIT	Git a jiné systémy pro správu verzí	7	TEMNO	Teorie množin a matematika nekonečen	11
GIT2	Git pro pokročilé	7	NONEX	Teorie nemožného	12
HAUS	Hausdorffův zvěřinec	12	NA	Teorie sítí	12
IPV6	IPv6	9	NUT2	Teorie čísel a RSA	11
CPU	Jak postavit (rychlý) procesor	8	QA	Testování a kvalita softwaru	8
STYLE	Jak se nestat vepřem	8	VIM	Textový editor Vim	8
JS	JavaScript	7	TOP	Top-stromy	6
GOLANG	Jazyk Go	6	TYPO	Typografie	10
RUST	Jazyk Rust	7	SLOZ3	Třídy složitosti	10
CPLX	Komplexní a komplexnější čísla	11	ANDINT	Vnitřnosti Androidu	9
ZIP	Komprese dat	11	VYCIS	Vyčíslitelnost	10
			HTTP	Web uvnitř	9

REACT	Webové UI funkcionálně	7	CPP	Černá magie v C++	7
ANDROID	Základy psaní pro Android	7	BAR	Čárové kódy	10
TEX	\TeX	10			
MA	Úvod do matematické analýzy	12			

Půlnoční přednášky

ANCAP	Bezstátní společnost	13	ORI	Orientace	13
FON	Fonetika	13	SCIFI	Technika ve sci-fi	13
HIPPO	Hippo::Web	13	ZDRAV	Základy první pomoci	13
LING	Lingvištika	13	TEA	Čaj	13
DRON	Létání a natáčení s dronem	13			
ORG	Organizování	13			