Návrh řešení složitější úlohy – upřesnění zadání; postřehy & problémy; zdůvodněný výběr algoritmu (vyjmenovat i další, složitost); návrh datových struktur (co v paměti, co na disku; pole, halda, BVS...); dekompozice; diskuze (kdyby...)

Programování – schopnost řešit úlohy; umění psát programy
Program je předpis, podle kterého může počítač provádět výpočet algoritmu
Algoritmus – k řešení úloh; výsledek lze získat mechanicky; konečnost

Dokázání konečnosti
Ověření správnosti – korektnost, neopomenout žádnou možnost
- částečná správnost – skončí-li, dá správný výsledek
- invariant – tvrzení platící pro celou dobu
- kromě netypických výjimek nikdy nestačí k ověření správnosti provést konečný počet výpočtů

Složitost algoritmu – časová, paměťová složitost v závislosti na velikosti vstupních dat
- asymptotická složitost
- který algoritmus je nejrychlejší závisí na vstupních datech

Složitost úlohy – složitost nejlepšího algoritmu, který úlohu řeší
Mám-li nějaký algoritmus, zjistím jeho složitost a poté hledám zrychlení

Podprogramy zkrátí nebo zpřehlední program, možnost práci rozdělit
Správná hodnota aritmetických funkcí je zaručena pouze tehdy, jsou-li všechny mezivýsledky zobrazitelné
Každá rekurze musí mít podmínku, aby se nevolala donekonečna

Fronta - seznam prvků, vybírám první vložený
Zásobník - seznam prvků, vybírám poslední vložený

Virtuální metoda – polymorfismus – volá se taková metoda, jaké instance je objekt; Tabulka VM
Statická metoda – adresa známá už při překladu; svázány s objektem, u následníků/ předků nemají smysl

Strukturované programování
Algoritmus se rozděluje na dílčí úlohy, které se spojují v jeden celek; nepoužívá se příkaz skoku.
Programování zhora
Celkový problém rozdělit na části a ty postupně zkoumat
Rozděl a panuj
Rozdělení úlohy na menší části nad nimiž se provádí algoritmus (např. rekurzivně)
Dynamické programování
Řešení úlohy se zkládá z řešení stejné úlohy menšího rozměru (úlohy se mohou překrývat)

Objektově orientované programování
Jednotlivé prvky jsou seskupeny do objektů, které si pamatují stav a na venek poskytují operace
Objekty se na venek chovají jako černé skříňky, není třeba znát jejich vnitřní chování
Objekt nemůže přistupovat k vlastnostem jiných objektů, jen tak, jak mu to objekt dovolí – přes jeho rozhraní
Objekty seskupeny stromovou kulturou, mohou po sobě dědit
Různé objekty mohou mít stejnou metodu definovanou různě, u dědění mohu volat metodu předka

Událostmi řízené programování
Program ve spuštěném stavu čeká na nějakou událost (stisk klávesy... - zpracuje operační systém a pošle programu pokyn), v okamžiku kdy přijde, provede příslušnou akci a čeká dál (pokud to nebyl pokyn k ukončení programu)
Program zahájí činnost nastavením vnitřního stavu a dále je řízení předáno smyčce události, kde čeká na akci. Zprávy o těchto událostech jsou pak přeposílány dalším částem programu, které je zpracovávají.

Model, simulace
Vždy zjednodušuje skutečnost; simulační *t*; cíl není optimalizovat, ale zjišťovat závislosti na změně parametrů
Spojité vs. Diskrétní – v pravidelných časových intervalech sleduji model vs. Simuluji jen to, kde se něco děje

Hry

Programování konečných her hraných dvěma hráči, s úplnou informací (oba hráči mají všechny informace)
Alespoň pro jednoho hráče existuje neprohrávající strategie
Strom hry – všechny možné průběhy

- Vyhrávající – existuje tah, kdy může aktuální hráč vyhrát
- Prohrávající – neexistuje tah vedoucí do vyhrávající pozice

Reprezentace grafu

Seznam hran a separovaných vrcholů – pro řídké grafy
Matice sousednosti – n^2 paměť; okamžitě, zda je hrana + její cena
Matice incidence – mxn; d_{ik}=-1 – i je začátek hrany e_k, 1 – i je konec hrany e_k, 0 – jinak
Seznam následníků

Vyhledávací stromy

Červenočerné stromy - ...
AVL - ...
B-stromy - …

Hashování

Zřetězením – spojové seznamy
Otevřené adresování – případné přepočítávání náhradních pozic

Algoritmy:

Zvyšování efektivity: Předvýpočet
Chytrá rekurze – úkládání vypočtených hodnot do pomocného pole
Ořezávání – vyhodnocujeme každý uzel a neperspektivní možnosti nezkoumáme
Heurestika – odhad, kde je největší šance najít řešení, dle toho pořadí

Algoritmus vlny (prohledávání do šířky)
Na kolikátý krok se dostaneme figurkou na určitou polohu na šachovnici
Po vlnách přiřazuji hodnoty, kdy tato situace může nastat a pak kouknu, kdy nastal můj případ...

Algoritmus prohledávání s návratem (backtracking; prohledávání do hloubky)
Chodby zeleně, projde-li jednou, žlutě, projde-li dvakrát, červeně
více než jedna žlutá, zpět; je-li nějaká zelená, jde po ní; žlutá, zpět; výchozí bod, nelze splnit rozkaz

Hledání druhého nejmenšího prvku
Najdu nejmenší a pak porovnám jen ty, které s ním prohrály.

Hornerovo schéma
Přečteme poslední znak, převedeme na číslo, vynásobíme deseti, a přičteme na číslo převedený další znak...

Eratosthenovo síto
Chci-li provčísla od 2 do N, vypíšu si čísla 2, …, N
v každém kroku vezmu nejnižší provočíslo a všechny násobky vyškrtám

Euklidův algoritmus
Hledání největšího společného dělitele – větší číslo nahradím (větší mod menší), když menší=0, větší je NSD

Obecný algoritmus prohledávání
Do seznamu vlož výchozí pozici
dokud seznam není prázdný a nebyl nalezen cíl
vyber a odstraň prvek ze seznamu
přidej do seznamu všechny dosud neprozkoumané prvky dostupné z aktuálního jedním krokem
Použiji-li frontu => prohledávání do šířky – algoritmus vlny
Použiji-li zásobník => prohledávání do šířky – algoritmus prohledávání s návratem
Prohledávání pole se zarážkou - na konec pole přidám hledaný prvek a při nálezu testuji, zda to není ten přidaný
Hledání k-tého nejmenšího prvku – Quicksort, počítám, kolik jich je na které straně a tu pak zkoumám. n až n^2

MiniMax
Strom průběhu hry, ohodnocení listů, střídavě po vrstvách vybíráme minimální/ maximální z těch dvou pod vrcholem – podle toho, zda vybírám já, nebo soupeř

NegaMax
jako minimax, akorát nevybírá minimum ale maximum z prvků vynásobených -1 (lze použít jednu proceduru)

AlfaBeta
Hledám-li v nějaké úrovni maximum, mám dočasnou hodnotu z jednoho syna a v dalším synovi dosáhne soupeř minima nižšího, než to mé maximum, už dál tu větev neprocházím, protože to neovlivní... (a naopak)

Burza
Zvolím nějakou cenu a sleduji, zda je větší nákup/ prodej, dle toho upravím... měním do požadované přesnosti.

Scrabble
Vytvořím dopředný a zpětný spoják; a pro každou skupinu písmen koukám, zda by to tím mohlo začínat/ končit

Tok v sítích
Algoritmem vlny najdu nejkratší spojení dvou bodů, hranu s minimální kapacitu „zruším“ a hledám znovu“

Třídící algoritmy

Složitost třídění O($n$log$n$) – zhora ohraničuje MS... zdola $n!$ vstupů -> $n$log$n$ větvení rozhodovacího stromu

Vnitřní třídění – všechna data načtena do pamětí; vnější třídění – v paměti jen část

Přímým výběrem: Z nesetříděné části vezmu nejmenší prvek a přesunu ho na konec té setříděné
O(n^2) přesunů, n^2 porovnání

Přímým vkládáním: Vezme prvek na řadě a vloží ho do setříděné části
O(n^2) přesunů, $n$log$n$ porovnání

Bublinkové třídění (Bubblesort): Projede pole a každé dva prvky ve špatném pořadí přehodí
O(n^2) prohození

Třídění přetřásáním: Jednou prochází zleva, jednou zprava, pamatovat si meze, kde se něco přerovnávalo

Shellovo třídění: Nejdřív budu koukat na prvky daleko od sebe
Časová složitost nebyla odvozena, na základě experimentů O($n\log^2 n$)

Halda (Heapsort): Binární strom, všechny vrstvy až na poslední zaplněny, poslední zaplněna zleva
každý otec je menší než jeho synové
Vložíme prvek nakonec a necháme probublat; pak od listů vždy slejeme seznamy se společným otcem
O(n) (vytvoření haldy) O(log$n$) (vložení prvku)
Postavit haldu z N prvků; je-li další prvek menší než minimum, vypíšu haldu, jinak jen minimum
Postavím haldu; přijde-li větší prvek, vypíšu *min* a zapojím; jinak vypíšu *min* a zapojím prvek do druhé

Quicksort: Rozdělím pole na dvě části; přiřazení dle toho, zda je prvek menší/ větší než pivot
O(n^2)

Mergesort (třídění sléváním): Dva setříděné systémy, v obou jsem na nejmenším prvku a ten menší přidám
O($n$log$n$)
Přímé sloučování rozdělím bloky podle násobků dvou...
Přirozené – vezmu jednotlivé souvislé bloky...
Dvoufázové - data v jednom souboru za sebou, v prvním kroku rozdělit do dvou souborů, ve druhém slít
Jednofázové – bloky ukládány střídavě do jednoho a do druhého souboru
Polyfázové – Dva soubory, v jednom F_{n+1} bloků, v druhém F_n bloků, po slití F_n a F_{n-1} bloků (zbyly)

Radixsort: Třídí podle jednotlivých cifer, od nejméně významné, zachovávat pořadí (např. spojové seznamy)
O($d(n+p)$) – n d-místných čísel, p přihrádek

Grafové algoritmy

Minimální kostra – hladový algoritmus

Nalezení nejkratší kostry
z jednoho vrcholu – Dijskritův

Souvislost grafu: pro každou hranu kouknu zda jsou vrcholy v různých komponentách, pak je spojím

Dijsktrův algoritmus: Orientovaný graf s nezáporným ohodnocením, cesty z jednoho vrcholu
Z nenavštívených vrcholů vyberu ten s nejmenším ohodnocením, přidám k navštív., přepočítám nenav.

Bellman-Fordův algoritmus: Zda obsahuje nezáporný cyklus z v. V každém kroku přepočítám všechny hrany

Minimální kostra grafu: Vyberu min. hranu a pokud s ní nevytvořím cyklus, přidám ji a spojím komponenty

Floyd-Warshallův algoritmus: Matice, pro každý vrchol zkusím, zda je cesta přes tento vrchol kratší

Topologické uspořádání: uspořádání $v_1, v_2, \ldots, v_n$, takové, že pro všechny hrany (v_i, v_j) platí $i<j$
Graf je acyklický; Prohledávání do hloubky nenajde zpětnou hranu (zpětnou, ne příčnou ;-)