Rekurze

Objekt je soucasti sebe samotneho. Můžeme si to predstavit jako metodu, ktera na konci zavolá sama sebe, tim padem se uzavre do cyklu. Je dobry mit nejakou funkcni, po kterem se splni a tim padem nastal nejaky definitivni konec rekurzivni metody. Pri praci s rekurzi se snazime program co nejvice optimalizovat, rekurze dokaze zrat hodne pameti

```
def soucet_cisel(n):  2 usages
    if n == 1:
        return 1
    else:
        return n + soucet_cisel(n - 1)

n = 5
result = soucet_cisel(n)
print(result)
```

Rekurze je vhoda pro prochazeni stromu a grafu, implementace algoritmu rozdel a panuj, hledani nejkratsi cesty v grafu

Typy rekurze
Přímá - funkce volá sama sebe, nejběžnější forma
Nepřímá - funcke A volá funkci B, která volá opět A, volaj se v kruhu

Brute Force
Brute force je technika prochazeni, ktera pouziva hrubou silu - jsou postupne prochazeny vsechny prvky bez predem dane logiky. Jedna se vyhradne o deterministicky algoritmus.

deterministicky - vraci vzdy stejny vysledek
nedeterministicky - nevraci vzdy stejny vysledek

Je extremne casove a pametove narocny, asymptoticka slozitost je **O(n!)**.Cas potrebny pro prolomeni roste s delkou klice, kterou uvadime v bitech. Cim delsi klic je, tim to bude trvat brute forcu vyresit a celkove sifra je bezpecnejsi.
Na druhou stranu je velmi trivialni a v kazdem pripade u kazdeho problemu najde vysledek, i kdyz to muze trvat extremne dlouho

Vyuziva se k prolomovani hesel. Mame pin o 5 prvcich, tim padem mame na kazdy jeden presne 9 prvku (10x10x10x10x10=100 000). Brutre force projde vsechny moznosti, s tim ze vi, ze jedna kombinca je 100% spravne.

Heuristika
Heuristika je způsob řešení problému pomocí zkušeností a intuice, když nemáme přesný algoritmus. Jde o rychlé nalezení dostatečně dobrého řešení, ne nutně optimálního. Heuristiky se používají, když by přesné řešení trvalo příliš dlouho nebo když nemáme všechny informace.

Heuristické algoritmy najdou řešení složitých problémů rychleji než metody hrubé síly, ale bez záruky nalezení optimálního řešení. Můžou uvíznout v lokálních optimech a jejich úspěšnost závisí na kvalitě heuristiky.

Navigace v autě představuje klasický příklad heuristiky. Při cestě z Prahy do Brna navigace nenabídne úplně všechny možné trasy, což by vyžadovalo ohromné výpočetní zdroje a trvalo příliš dlouho. Místo toho aplikace používá heuristiku: vybere dálnici D1, protože je to obvykle nejrychlejší spojení, i když ne vždy ideální (například při dopravních zácpách).GPS neprochází systematicky všechny miliony možných odbočení na každé křižovatce mezi oběma městy. Používá zjednodušující pravidla jako 'preferování hlavních silnic' a 'směřování přibližně k cíli'. Takové řešení lze najít okamžitě a je dostatečně kvalitní pro praktické použití, přestože teoreticky by mohla existovat nějaká méně známá trasa, která by mohla být o několik minut rychlejší.

Nedeterministicke algoritmy

deterministicky - za stejnych podminek vraci vzdy stejny vysledek
nedeterministicky - za stejnych podminek nevraci vzdy stejny vysledek

Nedeterministické algoritmy často využívají náhodné generování čísel, pravděpodobnostní rozhodování nebo paralelní zkoumání několika možných cest řešení současně.

Priklad Monte Carlo algoritmus
Monte Carlo je algoritmus, ktery pracuje jenom s urcitou asti dat. Mejme 100 prvkovou mnozinu, Monte Carlo vybere nekolik nahodne prvku a tema pracuje. Tim padem kdyz vzdy vybira jina data, tak vyjde jiny vysledek