

Rekurze

Objekt je součástí sebe samotného. Můžeme si to představit jako metodu, která na konci zavolá sama sebe, tím pádem se uzavře do cyklu. Je dobrý mít nějakou funkci, po které se splní a tím pádem nastal nějaký definitivní konec rekurzivní metody. Při práci s rekurzi se snažíme program co nejvíce optimalizovat, rekurze dokáže ztratit hodně paměti

```
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 vhodná pro procházení stromu a grafu, implementace algoritmu rozdel a panuj, hledání nejkratší cesty v grafu

Typy rekurze

Přímá - funkce volá sama sebe, nejběžnější forma

Nepřímá - funkce A volá funkci B, která volá opět A, volají se v kruhu

Brute Force

Brute force je technika procházení, která používá hrubou sílu - jsou postupně procházeny všechny prvky bez předem dané logiky. Jedná se výhradně o deterministický algoritmus.

deterministický - vrací vždy stejný výsledek

nedeterministický - nevrací vždy stejný výsledek

Je extrémně časově a paměťově náročný, asymptotická složitost je $O(n!)$. Čas potřebný pro prolomení roste s délkou klíče, kterou uvádíme v bitech. Čím delší klíč je, tím to bude trvat brutě forcí vyřešit a celkově síla je bezpečnější.

Na druhou stranu je velmi trivialní a v každém případě u každého problému najde výsledek, i když to může trvat extrémně dlouho

Využívá se k prolomování hesel. Máme pin o 5 prvcích, tím pádem máme na každý jeden přesně 9 prvků ($10 \times 10 \times 10 \times 10 \times 10 = 100\,000$). Brutě force projde všechny možnosti, s tím že ví, že jedna kombinace je 100% správná.

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 'směrování 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 stejných podmínek vrací vždy stejný výsledek

nedeterministicky - za stejných podmínek nevrací vždy stejný výsledek

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

Příklad Monte Carlo algoritmus

Monte Carlo je algoritmus, který pracuje jenom s určitou sítí dat. Mějme 100 prvkovou množinu, Monte Carlo vybere několik náhodně prvků a na nich pracuje. Tím pádem když vždy vybírá jiná data, tak vyjde jiný výsledek