Řešení úlohy č. 5
Věže

Časová složitost přesunu z A na B

Časová složitost prvního algoritmu je $O(2^n)$. Pro věž výšky n platí, že je potřeba udělat

$$T(n) = 2T(n-1) + 1$$

přesunů, kde +1 značí přesun nejnižšího disku z A na B a $T(n-1)$ znamená dočasný přesun všech $n-1$ disků na C a jejich přesun zpět na A. Lze si povšimnout, že největší disk se přesune pouze jednou. Disk nad ním se dvakrát přesune v rámci přesunu nejnižšího disku a pak jednou při přesunu na cílové B a tak dále. Jinými slovy, každým diskem pohybujeme minimálněkrát a proto je tento algoritmus optimální. Ukázat, že $T(n) \in O(2^n)$ lze následující úpravou:

$$\begin{aligned}
T(n) &= 2T(n-1) + 1 \\
T(n) &= 2(2T(n-2) + 1) + 1 \\
T(n) &= 2^2 T(n-2) + 2^1 + 1 \\
&\vdots \\
T(n) &= 2^k T(n-k) + 2^{(k-1)} + \ldots + 2^0 \\
&\vdots \\
T(n) &= 2^n - 1
\end{aligned}$$

a $T(0) = 0$, protože když nemáme disky, tak není co dělat.

Paměťová složitost je $O(n)$, protože si kromě pozic jednotlivých disků není co pamatovat. Navíc bude zanoření rekurze pouze tolik, kolik máme disků, protože v každém kroku rekurze se funkce volají na část věže, které jsou alespoň o jedno patro nižší. Toto bude platit i v následujících úlohách.

Algoritmus pro přesun z A na B bez přímého přesunu A na B

Ukažme si nejprve pseudokód:

```
1 přesuň(koho, odkud, kam, přes):
2   pokud je koho prázdné, pak skonči
3   jinak:
4     použij přesuň(celou věž kromě nejnižšího disku, odkud, kam, přes)
5     přesuň jediný disk v "odkud" na "přes"
6     použij přesuň(celou věž, kam, odkud, přes)
7     přesuň jediný disk v "přes" na "kam"
8     použij přesuň(celou věž, odkud, kam, přes)
```

Časová složitost tohoto algoritmu lze vyjádřit obdobně vzorcem

$$T(n) = 3T(n-1) + 2.$$

Optimalita opět vyplývá z minimality počtu přesunů nejnižšího disku. Rekurentní vztah lze upravit obdobně a dosáhneme časové složitosti $O(3^n)$.

Prohození věží

Pro účely této sekce předefinujeme mírně předefinujeme funkci *přesuň* ze zadání následujícím způsobem:

```
1 přesuň(kolik, odkud, kam, přes):
2   pokud kolik < 1:
3     nedělej nic
4   jinak:
5     použij přesuň(kolik - 1, odkud, přes, kam)
6     přesuň jeden disk z odkud na kam
7     použij přesuň(kolik - 1, přes, kam, odkud)
```

Nastaly 2 změny, základním krokem rekurze ve funkci *přesuň* je nyní prázdná věž a parametr pro určení počtu přesouvaných disků předáváme číselně místo textového popisu. Jinak funkce zůstává stejná (včetně funkčnosti a složitosti).

Světy pojmenujme jako světelný, stínový a prázdný podle typu disků, který se ve světech nachází na začátku úlohy. Označme počet disků v jedné věži jako n, celkový počet disků je tedy $2n$. Dále BÚNO předpokládejme, že světy jsou umístěné zleva doprava v pořadí světelný, stínový a prázdný.

V první části chceme obě věže sloučit do jedné. Chceme vždy vybrat buď světelnou nebo stínovou věž a přesunout na druhou z nich co nejvíce disků tak, aby se neporušily žádné podmínky v zadání. Lze si snadno rozmyslet, že tento postup budeme provádět maximálně tolikrát, kolik je disků, protože v každé iteraci se přesune alespoň 1 disk. Přesunu lze dosáhnout funkcí *přesuň* ze zadání. Aby funkce *přesuň* fungovala, tak musí být zaručeno, že svět *přes* je prázdný při každém volání. To, že v cílovém světe nepřekáží disk, který by zabránil přesunu je zaručeno způsobem, kterým vybíráme počet disků k přesunutí a podmínka ze zadání která říká, že světelná věž a stínová věž jsou až na světelnost identické, zejména co se týče velikostí disků v odpovídajících si patrech. Roli světa přes vždy hraje *prázdný* svět, ve kterém si nic neodkládáme mezi voláními funkce *přesuň*. Tento postup zapišme do pseudokódu:

```
1 pro i od 1 do 2n včetně s krokem 1:
2   pokud je i dělitelné 2:
3     použij přesuň(i, stínový, světelný, prázdný)
4   jinak:
5     použij přesuň(i, světelný, stínový, prázdný)
```

V poslední iteraci se celá věž přesune do svetelného světa tak, že v přízemí bude stínový disk. Kromě toho pouze víme, že věže jsou sloučené tak, že stejně velké disky jsou u sebe. Pro dvojici disků velikosti k nedokážeme říct, jestli je výše světelný nebo stínový disk, nic nevíme. To nám však vůbec nevadí a rozložení lze stále provést. Stačí nyní v každé iteraci zkontrolovat, jestli nejspodnější disk právě přesouvané podvěže už není náhodou na správném místě. Pokud je, tak přejdeme k další iteraci. V prázdném světě nic mezi voláními neodkládáme, přesuny funkcí *přesuň* lze realizovat. Tento postup opět zapišme do pseudokódu.

```
1 pro i od (2n - 1) do 1 včetně s krokem -1:
2   pokud je i dělitelné 2:
3     je-li i-tý disk stínový:
4       pokračuj na další iteraci
5     použij přesuň(i, stínový, světelný, prázdný)
6   jinak:
7     je-li i-tý disk světelný:
8       pokračuj na další iteraci
9     použij přesuň(i, světelný, stínový, prázdný)
```

Proveďme analýzu časové složitosti obou funkcí najednou, protože jsou téměř identické. Víme, že časová složitost funkce *přesuň* pro věž výšky n je $2n - 1$. Z toho nám vyplývá následující vztah pro časové složitosti:

$$T(n) = \sum_{k=1}^{2n} 2^k - 1.$$

Jelikož je funkce $T(n)$ rostoucí, pak její součet dokážeme odhadnout následujícím vztahem:

$$f(1) + \int_1^n f(x)dx \leq \sum_{k=1}^{n} f(k) \leq f(n) + \int_1^n f(x)dx.$$

Spočtěme tedy horní odhad

$$\int_1^n 2^x - 1dx = \left[\frac{2^x}{\ln(2)} - x\right]_1^{2n} = \frac{4^n}{\ln(2)} - 2n - \frac{2}{\ln(2)} + 1$$

z čehož dostáváme

$$\sum_{k=1}^{2n} 2^k - 1 \in O(4^n).$$

Časová složitost obou naších funkcí je tedy $O(4^n)$, celý algoritmus tedy běží v čase $O(4^n)$, kde n je celkový počet disků v obou věžích.