DETYRA E PARË

Krijimi i një aplikacioni që zgjidh problemin e Kullës së Hanoit me anë të metodave:

1. ***Rekursive***
2. ***Iterative***

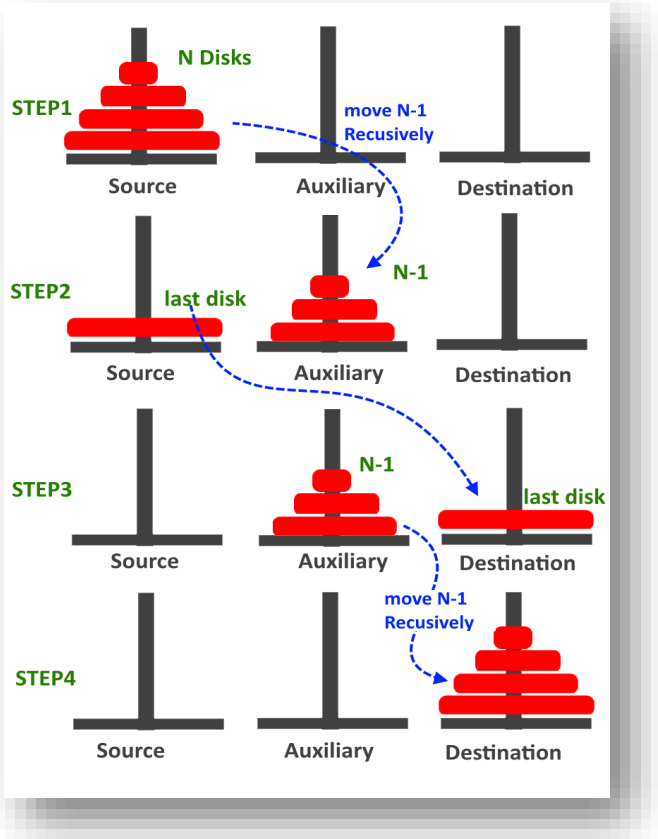
<u>Përshkrimi i enigmës: Kulla e Hanoit</u>

Kulla e Hanoi është një enigmë matematikore ku kemi tre shufra (A, B dhe C) dhe N disqe.

Fillimisht, të gjithë disqet grumbullohen në një vlerë në rënie të diametrit, d.m.th., disku më i vogël vendoset në majë dhe ata janë në shufrën A.

Objektivi i enigmës është të zhvendosë të gjitha disqet në një shufër tjetër (këtu konsiderohet C), duke iu bindur rregullat e mëposhtme të thjeshta:

1. ***Vetëm një disk mund të zhvendoset në të njëjtën kohë.***
2. ***Çdo lëvizje konsiston në marrjen e diskut të sipërm nga njëra prej shufrave dhe vendosjen e tij mbi një shufër tjetër, d.m.th. një disk mund të zhvendoset vetëm nëse është disku më i lartë në një shufër.***
3. ***Asnjë disk nuk mund të vendoset në majë të një disku më të vogël.***



<u>Algoritmi Rekursiv:</u>

Pseudokodi:

```
function towerOfHanoi(n, from, to, aux)
   if n==0 then
     return;
   end
   towerOfHanoi(n-1, from, aux, to);
   print("Move disk " + n + " from "+
from + " to " + to);
   towerOfHanoi(n-1, aux, from, to);
end
```

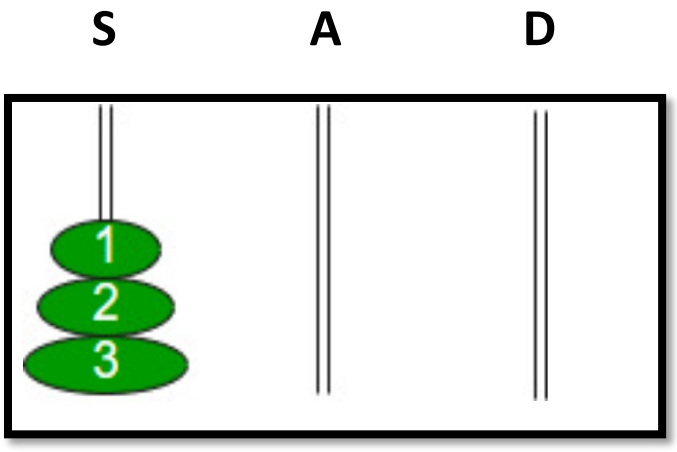
Hapat e zgjidhjes përmes Algorimit Rekursiv:

1. ***Krijojmë një funksion towerOfHanoi që përmban variablat N (numri aktual i diskut), from, to, aux.***
2. ***Pastaj thirrim funksionin për diskun e (N – 1).***
3. ***Pastaj printojmë diskun aktual së bashku me from dhe to***
4. ***Përsëri thërrasim funksionin për diskun e (N – 1).***

Për ta optimizuar zgjidhjen, ne kemi shfrytëzuar metodën iterative, ku përmes for loop, while loop ose loop-ën do-while kemi përsëritur të njëjtat hapa, në vend të thirrjes së vet metodës, ashtu siq realizon Algoritmi Rekursiv, I cili ngadalëson procesin duke thirrur vetvetën për çdo ekzekutim.

Algoritmi Iterativ

1. **Derisa n të bëhet 1, do të vendosim një variabël në stack e cila bën një gjurmë të polit burimor, ndihmës dhe destinacionit.**
2. **Nëse n bëhet 1, atëherë ne do ta nxjerrim variablën nga stack dhe do ta printojmë atë.**



Le të kuptojmë me një shembull të thjeshtë me 3 disqe:

1. **Pra, numri i përgjithshëm i lëvizjeve të kërkuara = 7, dhe Kur i= 1, (i % 3 == 1) lëvizja e lejuar realizohet ndërmjet ‘S’ dhe ‘D’**
2. **Kur i = 2, (i % 3 == 2) lëvizja midis 'S' dhe 'A'**
3. **Kur i = 3, (i % 3 == 0) lëvizja ndodh midis ‘D' dhe ‘A'**
4. **Kur i = 4, (i % 3 == 1) lëvizja e lejuar ndodh midis 'S' dhe 'D'**
5. **Kur i = 5, (i % 3 == 2) lëvizja e lejuar midis 'A' dhe ‘S'**
6. **Kur i = 6, (i % 3 == 0) lëvizja e lejuar midis 'A' dhe 'D'**
7. **Kur i = 7, (i % 3 == 1) lëvizja e ligjshme midis 'S' dhe 'D’**

Pra, në fund të fundit, këto pole të destinacionit përmbajnë të gjitha sipas rendit të madhësisë.

Pas vëzhgimit të përsëritjeve të mësipërme, mund të mendojmë se pasi të zhvendoset një disk përveç diskut më të vogël, disku tjetër që do të zhvendoset duhet të jetë disku më i vogël, sepse është disku i sipërm që mbështetet në shtyllën e lirë dhe nuk ka zgjedhje të tjera për të lëvizur një disk.

Kompleksiteti kohor:

Time Complexity, apo kompleksiteti kohor, është numri i operacioneve që kryen një algoritëm për të përfunduar detyrën e tij (duke marrë parasysh që çdo operacion kërkon të njëjtën kohë).

$$
\begin{aligned}
f(n) &= f(n-1)+1+f(n-1) \\
&= 2f(n-1)+1 \\
&= 2(2f(n-2)+1)+1 = 2(2f(n-2))+2+1 \\
&= 2(2(2f(n-3)+1)+1)+1 = 2^3 f(n-3)+4+2+1 \\
&\vdots \\
&= 2(2(2(\cdots(2f(n-m)+1)\cdots)+1)+1)+1 \\
&= 2^m f(n-m)+2^{m-1}+2^{m-2}+\cdots+2^0 \\
&= 2^{n-1} f(1)+2^{n-2}+2^{n-3}+\cdots+2^0 \\
&= 2^n-1
\end{aligned}
$$

$O(2^n)$

Ka dy mundësi për çdo disk. Prandaj,

$2 * 2 * 2 * \ldots * 2$

(N herë) është 2^n

Efikasiteti i Algoritmeve

Algoritmi që kryen detyrën në numrin më të vogël të operacioneve konsiderohet më efikasi për sa i përket kompleksitetit kohor.

Gjatë testimit të kodit, aplikacioni ynë ka sjell rezultate se cili algoritëm ekzekuton të dhënat, dhe sjell rezultat më të shpejtë.

Rezultati i fituar është realizuar nga kodi i klasës Non-rekursive në të cilën është realizuar një matje e kohës së ekzekutimit për të dy metodat në formën e një tabele, ku paraqet ekzekutimin e tyre në sekonda, në milisekonda, në mikrosekonda, në nanosekonda, dhe në pikosekonda.