Analyse af algoritmer for ombytningspuslespil

Algoritmer for ombytningspuslespil

Prøv ombytningspuslespillet på kurset webside. Hvilken score opnåede du i tredie forsøg?

- Hvilken algoritme bruger du?
- Kan du sige noget om bedste og værste køretid for din algoritme for puslespil med n brikker?
- Er køretiden relateret til antal brikker, som står rigtigt til at starte med?
- Er den "grådige algoritme" ($=$ sæt én brik på plads i hvert skridt) bedst mulig, eller kan man få flere skridt hvor to sættes på plads ved nogle gange at undlade at sætte én på plads?
- Mere generelt, *kan vi præcist beskrive alle bedst mulige algoritmer?*

Model af puslespil

Vi modellerer brikkerne i et puslespil som tallene $1, 2, 3, \ldots, n$, nummereret efter den plads, brikken skal stå på:

5	10	14	3
1	11	9	15
8	7	2	12
4	13	6	16

$\rightarrow$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

NB: pladen kan også modelleres som et array/en liste (grå tal er indekser, her startende med 1):

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
5	10	14	3	1	11	9	15	8	7	2	12	4	13	6	16

Dette gør vi til opgavetimerne (i programmeringsopgaverne)

En opstilling af tallene $1, 2, 3, \ldots, n$ i et array af længde n kaldes også en *permutation*.

Den grådige algoritme og dens analyse

```
WHILE ikke alle brikker på plads:
  Vælg en brik ikke på plads
  Byt den med brikken på dens plads
```

For et puslespil med n brikker, hvad er det maksimale antal ombytninger?

> For hver ombytning: mindst én brik kommer på plads, og ingen brik forlader dens plads. Derfor maksimalt n ombytninger.

For et puslespil med n brikker, hvoraf t allerede står på plads, hvad er det maksimale antal ombytninger?

> For hver ombytning: mindst én brik kommer på plads, og ingen brik forlader dens plads. Derfor maksimalt $n - t$ ombytninger.

For et puslespil med n brikker, hvoraf t allerede står på plads, hvad er det minimale antal ombytninger?

> For hver ombytning: højst to brik kommer på plads. Derfor mindst $(n - t)/2$ ombytninger.

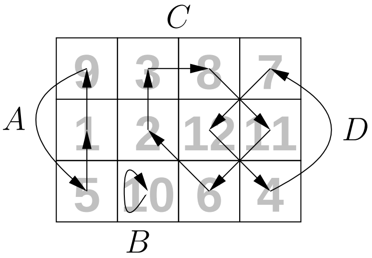
Kredse

Denne analyse på mellem $(n-t)/2$ og $n-t$ ombytninger er allerede ganske præcis (øvre og ned grænse er en faktor to fra hinanden).

Men vi kan lave en *endnu bedre* (mere præcis) analyse.

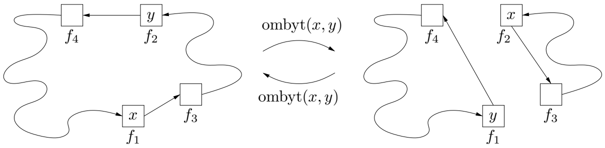
Observation: en permutation giver på naturlig måde anledning til en samling *kredse*:

Lad en brik (tal) t pege på den plads, hvor den skal stå, nemlig pladsen med index t.



Kredse og ombytninger

Observation: En ombytning af to brikker i samme kreds øger antal kredse med præcis én. En ombytning af to brikker i forskellige kredse mindsker antal kredse med præcis én:



Observation: Brik er på plads ⇔ brik er i en kreds af længde én.

Derfor: puslespil løst ⇔ der er n kredse.

Konklusion: Et puslespil med n brikker og k *kredse* i startopstillingen kræver mindst $n - k$ ombytninger, og man kan altid gøre det med $n - k$ ombytninger (bla. via den grådige algoritme, som i hvert skridt deler en kreds af længde t i to kredse af længde $t - 1$ og 1).

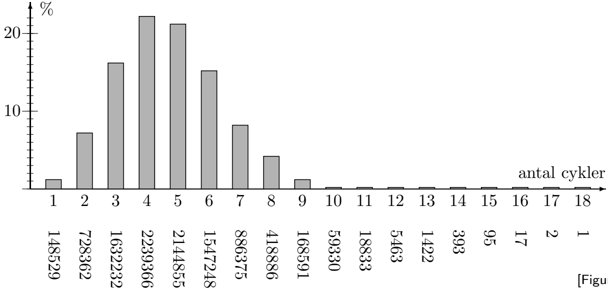
Konklusion: En algoritme bruger det optimale antal ombytninger $(n - k)$ hvis og kun hvis hver ombytning er med to brikker som er i samme kreds.

Forventet antal kredse

Sætning (uden bevis her): Det forventede antal kredse i en tilfældig permutation er

$$H_n = \sum_{i=1}^{n} 1/i$$

Ved simulering (afprøvning, 10.000.000 tilfældige permutationer) for $n = 64$ ses følgende fordeling af antallet af permutationer:



[Figur: Gerth Brodal]