DSK814: Algorithms and Data Structures

SDU - Kolding

Analysis of algorithms for swap puzzles

Analysis of algorithms for swap puzzles

Try the swap puzzle on the website.

https://cdn.tutsplus.com/active/uploads/legacy/tuts/403_html5TileSwapPuzzle/demo/puzzle.html

What score did you get on your third attempt?

Analysis of algorithms for swap puzzles

Try the swap puzzle on the course website.

https://cdn.tutsplus.com/active/uploads/legacy/tuts/403_html5TileSwapPuzzle/demo/puzzle.html

What score did you get on your third attempt?

- What algorithm do you use?

Analysis of algorithms for swap puzzles

Try the swap puzzle on the course website.

https://cdn.tutsplus.com/active/uploads/legacy/tuts/403_html5TileSwapPuzzle/demo/puzzle.html

What score did you get on your third attempt?

- What algorithm do you use?
- Can you say something about the best and worst running time of your algorithm for puzzles with **n** pieces?

Analysis of algorithms for swap puzzles

Try the swap puzzle on the course website.

https://cdn.tutsplus.com/active/uploads/legacy/tuts/403_html5TileSwapPuzzle/demo/puzzle.html

What score did you get on your third attempt?

- What algorithm do you use?
- Can you say something about the best and worst running time of your algorithm for puzzles with **n** pieces?
- Is the running time related to the number of pieces that are correctly aligned to start with?

Analysis of algorithms for swap puzzles

- Is the "greedy algorithm" (= placing one piece in its correct position at each step) the best possible, or can one achieve more steps where two pieces are placed correctly by sometimes avoiding placing one piece in its correct position?

Analysis of algorithms for swap puzzles

- Is the "greedy algorithm" (= placing one piece in its correct position at each step) the best possible, or can one achieve more steps where two pieces are placed correctly by sometimes avoiding placing one piece in its correct position?
- If the greedy algorithm is the best possible, are there also other algorithms that are just as good (or do you have to place one piece in its correct position every time to be the best possible)?

Analysis of algorithms for swap puzzles

- Is the "greedy algorithm" (= placing one piece in its correct position at each step) the best possible, or can one achieve more steps where two pieces are placed correctly by sometimes avoiding placing one piece in its correct position?
- If the greedy algorithm is the best possible, are there also other algorithms that are just as good (or do you have to place one piece in its correct position every time to be the best possible)?
- More generally,can we accurately describe all best possible algorithms?

Model of puzzle

We model the pieces of a puzzle as the numbers 1,2,3, . . . ,n, numbered according to the layout when the puzzle is solved:

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

Model of puzzle

We model the pieces of a puzzle as the numbers 1,2,3, . . . ,n, numbered according to the layout when the puzzle is solved:

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

→

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

Note: this can also be seen as an array/list:

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

A list of numbers 1,2,3, . . . ,n in an array of length n also called a **permutation**

Analysis of the greedy algorithm

```
WHILE not all pieces are in place:
    Select a piece not in place
    Swap it with the piece in its place
```

Analysis of the greedy algorithm

WHILE not all pieces are in place:
  Select a piece not in place
  Swap it with the piece in its place

For a puzzle with n pieces, what is the maximum number of swaps?
  For each swap: at least one piece comes into place, and no piece leaves its place.
  Therefore, a maximum of n swaps.

Analysis of the greedy algorithm

WHILE not all pieces are in place:
    Select a piece not in place
    Swap it with the piece in its place

For a puzzle with n pieces, what is the maximum number of swaps?

- For each swap: at least one piece comes into place, and no piece leaves its place.
- Therefore, a maximum of n swaps.

For a puzzle with n pieces, of which t already in place, what is the maximum number of swaps?

- For each swap: at least one piece comes into place, and no piece leaves its place.
- Therefore, a maximum of n-t swaps.

Analysis of the greedy algorithm

WHILE not all pieces are in place:
Select a piece not in place
Swap it with the piece in its place

For a puzzle with n pieces, what is the maximum number of swaps?
For each swap: at least one piece comes into place, and no piece leaves its place.
Therefore, a maximum of n swaps.

For a puzzle with n pieces, of which t already in place, what is the maximum number of swaps?
For each swap: at least one piece comes into place, and no piece leaves its place.
Therefore, a maximum of n-t swaps.

For a puzzle with n pieces, of which t already in place, what is the minimum number of swaps?
For each swap: at most two pieces will be in place.
Therefore at least $(n - t)/2$ swaps.

Cycles

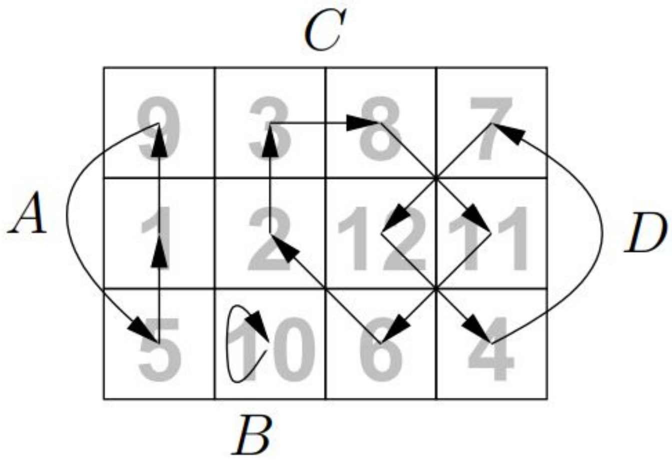
Can we do more? even better(i.e. even more precise) analysis than

“between $(n - t)/2$ and $n - t$ swaps”?

Cycles

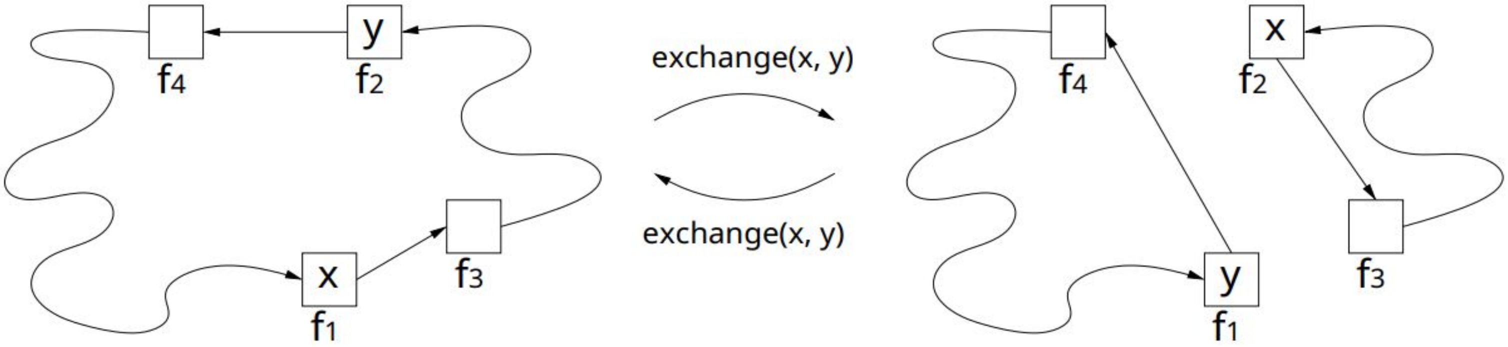
Can we do more? even better(i.e. even more precise) analysis than “between (n − t)/2 and n − t swaps”?

Observation: a permutation naturally gives rise to a collection of cycles:
A number (a piece)t points to the place where it should be, namely the place with number t.



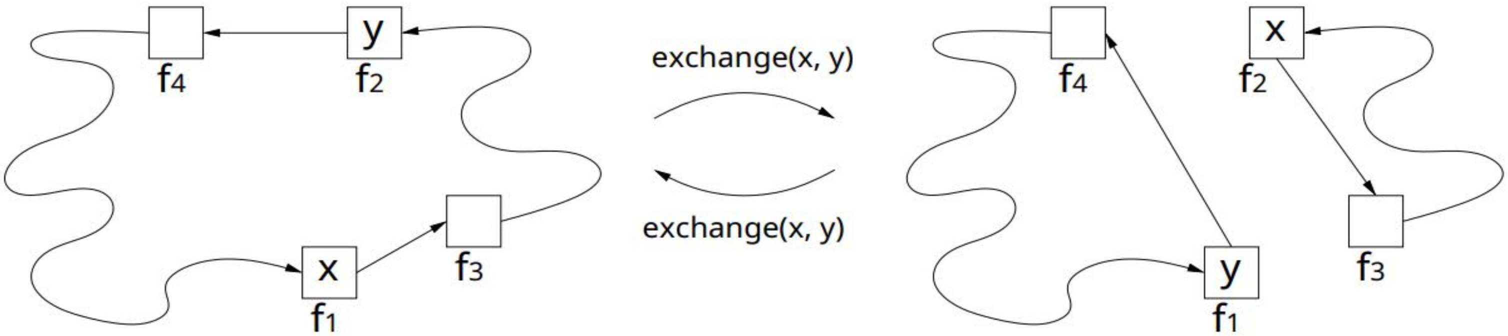
Cycles and swaps

Observation:
Swapping two pieces in the same cycles increases the number of cycles by exactly one. Swapping two pieces in different cycles decreases the number of cycles by exactly one:



Cycles and swaps

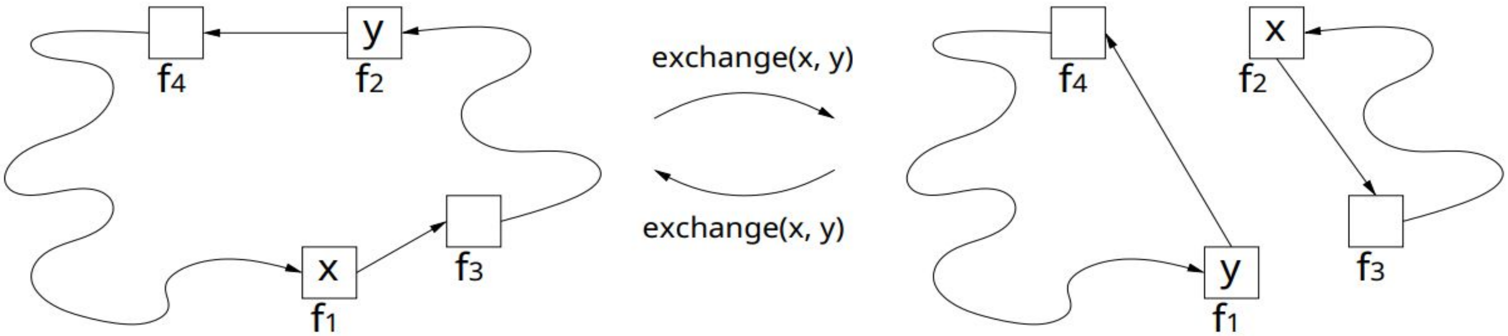
Observation:
Swapping two pieces in the same cycles increases the number of cycles by exactly one. Swapping two pieces in different cycles decreases the number of cycles by exactly one:



Observation: The piece is in place ⇔ piece is in a cycles of length one.

Cycles and swaps

Observation:
Swapping two pieces in the same cycles increases the number of cycles by exactly one. Swapping two pieces in different cycles decreases the number of cycles by exactly one:

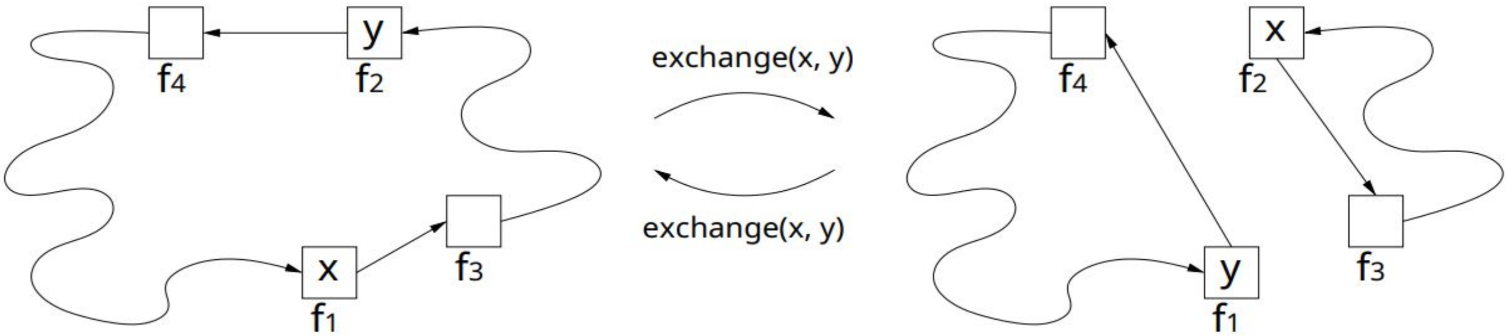


Observation: The piece is in place ⇔ piece is in a cycles of length one.

Therefore: puzzle solved ⇔ there is n cycles.

Cycles and swaps

Observation:
Swapping two pieces in the same cycles increases the number of cycles by exactly one. Swapping two pieces in different cycles decreases the number of cycles by exactly one:



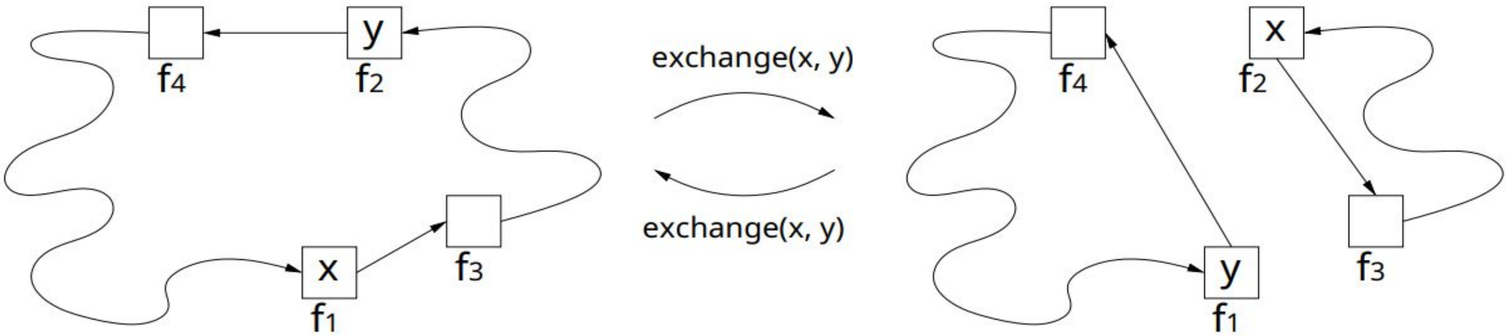
Observation: The piece is in place ⇔ piece is in a cycles of length one.

Therefore: puzzle solved ⇔ there is n cycles.

Conclusion: A puzzle with n pieces and k cycles in the initial setup requires at least n − k swaps, and it can always be done with n − k swaps (for example, using the greedy algorithm, which at each step splits a circle of length t into two cycles of length t − 1 and 1).

Cycles and swaps

Observation:
Swapping two pieces in the same cycles increases the number of cycles by exactly one. Swapping two pieces in different cycles decreases the number of cycles by exactly one:



Observation: The piece is in place.⇔piece is in a cycles of length one.

Therefore: puzzle solved ⇔ there is n cycles.

Conclusion: A puzzle with n pieces and k cycles in the initial setup requires at least n − k swaps, and it can always be done with n − k swaps (for example, using the greedy algorithm, which at each step splits a circle of length t into two cycles of length t − 1 and 1).

Conclusion: An algorithm uses the optimal number of swaps (n − k) if and only if each swap is with two pieces that are in the same cycles.

Expected number of cycles

Theorem (without proof here): The expected number of cycles in a random permutation is

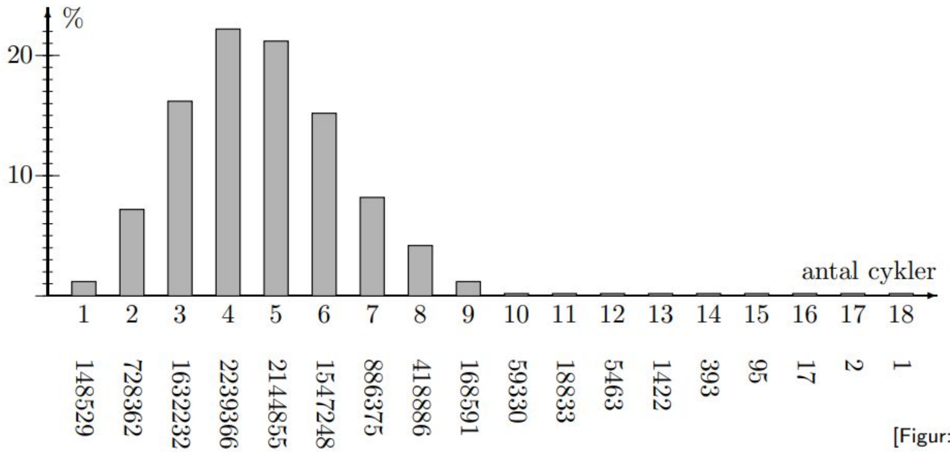
$$H_n = \sum_{i=1}^{n} 1/i$$

Expected number of cycles

Theorem (without proof here): The expected number of cycles in a random permutation is

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

With simulation (testing, 10,000,000 random permutations) for n = 64, the following distribution of the number of permutations is observed:



[Figur: Gerth Brodal]