ב״ה

Parallelism for Beginners with Fun Examples

- Assembling in parallel a 3D jigsaw puzzle ball.
- Parallel sorting of binary coded numbered cards.
- Using the twenty one card trick and its generalizations to motivate beginning students of Computer Science and Mathematics.

April 2019 - אדר ב׳ תשע״ט

Part 1 - Fun examples with parallel solutions

Assembling in parallel a 3D jigsaw puzzle ball.

Pieces are numbered on the reverse side.

Divide pieces into piles.

Pile of pieces numbered 1 to 9
Pile of pieces numbered 10 to 19
Pile of pieces numbered 20 to 29
Pile of pieces numbered 30 to 39
Etc.

Assemble pieces in each pile.

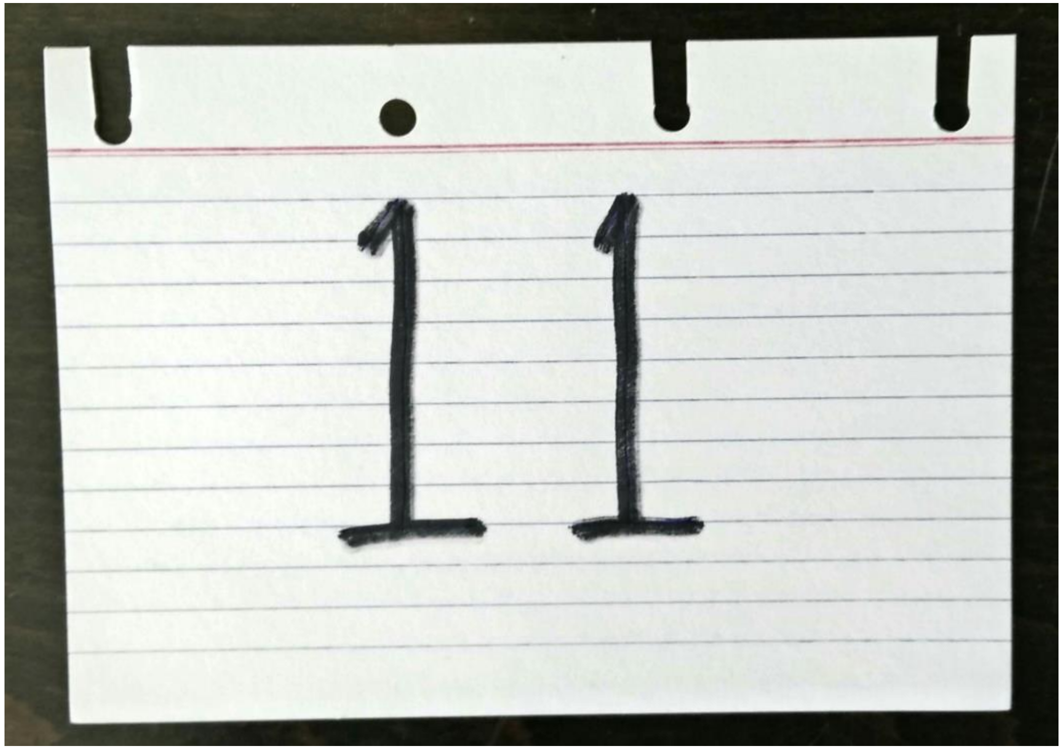
Join the partially assembled sections of the ball together.

Done in parallel (or sequentially).

Exercise / Question for discussion:
Adapt the above approach for sorting in parallel 16 cards, numbered 0, 1, 2, …, 14, 15, into increasing order.
Assume that four people participate in the work.

Sorting cards numbered from 0 to 15 into increasing order

Binary numbers are coded on the edge of the card.



VIDEO 1

This was shown to me when I was in school
without any mention of parallelism !

Exercises / Questions for discussion:

1) Explain why the cards get sorted?

2) How would you use this approach to sort cards into decreasing order?

3) Experiment to check that the sort works when some of the cards are missing or when there are duplicate cards.

4) How many holes/slots would be needed to sort cards numbered from 0 to 16?

5) What range of numbers may be sorted if there are 6 holes/slots?

6) How would you use this approach for parallel searching according properties?

Part 2 - Thinking prevents too much doing

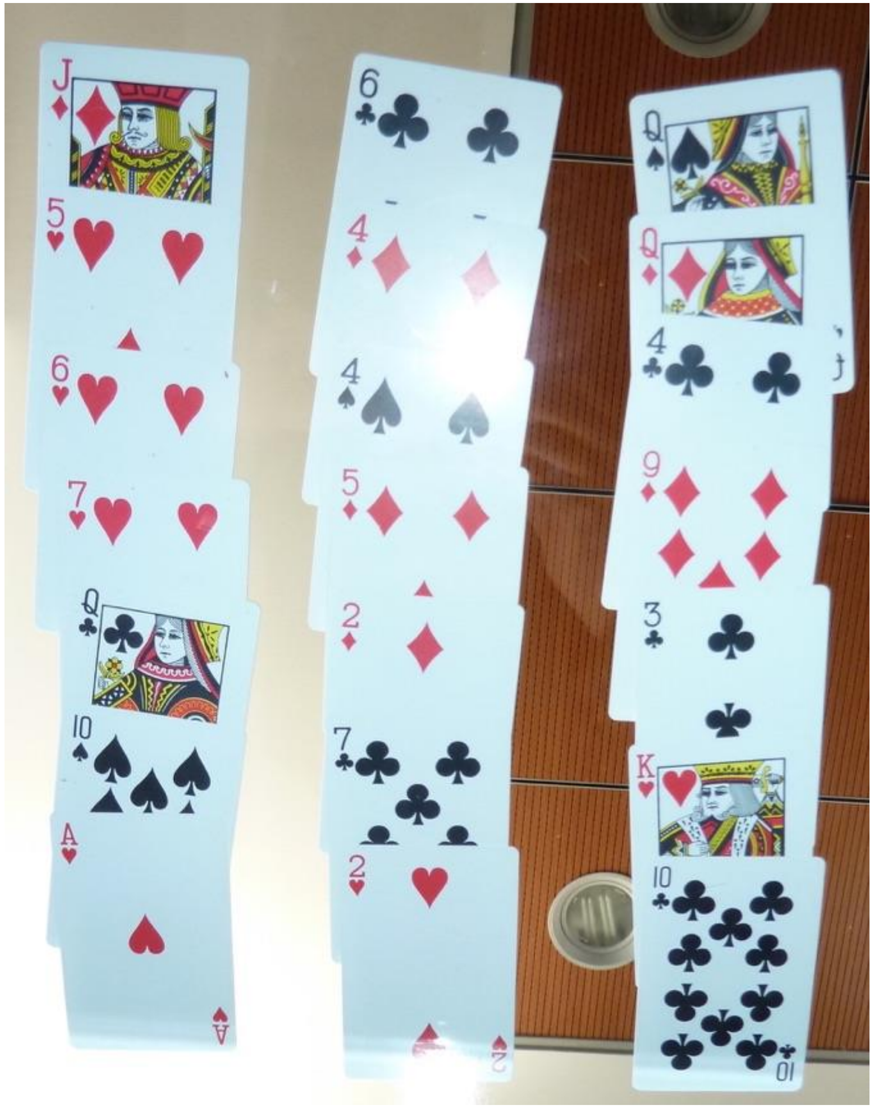
(My father’s first piece of advice ע״ה)

The twenty one card trick

Note that the table top is transparent

VIDEO 2

So, we managed to do something in parallel !



The twenty one card trick - summary:

Dealer and Chooser.

Cards dealt into rows and columns
(Omitted specific details, 7 rows by 3 columns.)

Cards dealt row by row.

The Chooser indicates the column of the chosen card.

If we don’t think enough, we may try all possibilities:

There are 21 cards and they can be arranged in 21! different arrangements. Therefore, the trick needs to be performed 21! times for all these arrangements to verify that the trick works in all cases.

21! = 51090942171709440000.

So let’s think a little !

Exercises / Questions for discussion:

1) (a) Check by a simple experiment, that the position of the chosen card after dealing, does not depend on the denominations of the cards.
   (b) Now explain why the position of the chosen card after dealing, does not depend on the denominations of the cards.

2) In view of this, how many times do we need to do the trick to check all possibilities?

3) Experiment to see if the trick works if the cards are dealt row by row, but when dealing rows, the cards are put down in various orders?

4) Experiment to see if the trick works if the cards are dealt into 3 rows by 7 columns? How many times do we need to ask: In which column is the card you chose?

Dealing the cards face down to understand why the trick works

VIDEO 3

Note that the 21 people can work in parallel.

Thinking some more:

The position in the column determines the outcome, but the actual column chosen has no effect on the outcome. This means the trick needs to be performed only 7 times and this may be done in parallel with 7 people and 7 packs of 21 cards.

The positions to be checked are from different rows.

Exercises / Questions for discussion:
Suppose you wished to check that the 7 rows by 3 columns card trick works in all cases. By dealing the cards face down, only 7 cases need to be checked.

1) What are these cases?

2) How would you organize the work if you did everything by yourself?

3) How would you organize the work for a team of 7 people?

4) How would you organize the work for a team of 3 people?

5) How would you organize the work for a team of 50 people?

6) Extra work needs to be done when working in parallel. Describe this extra work.
   Is it always worthwhile working in parallel?

7) Explain why the trick works if the cards are dealt row by row, but when dealing rows, the cards are put down in various orders?
   i.e. Parallel or unordered sequential dealing may take place when dealing the cards in a row.

8) *[Hard]* Find an example of statements which may be executed sequentially in any order giving a well-defined final result BUT has an undefined final result when executed in parallel?

Gathering as much information as possible

KEY:

"|" means card is vertical.

"–" means card is horizontal.

| – |

| – |

| – |

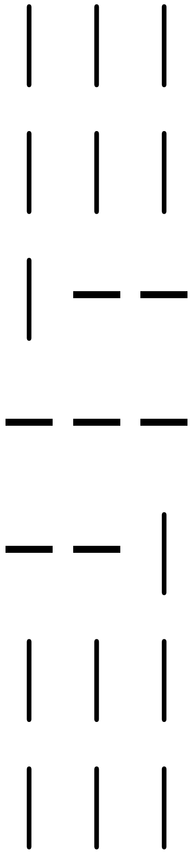
| – |

| – |

| – |

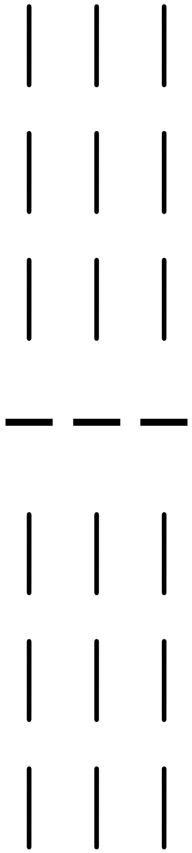
| – |

After dealing



Chosen cards of middle column includes those of other columns

Dealing as usual with the middle column above



Numbering the cards of the column chosen

□1□

□2□

□3□

□4□

□5□

□6□

□7□

The positions of the cards after dealing

□□□

□□□

□12

345

67□

□□□

□□□

The following table describes a function f(x) where x is the position of the card in the chosen column and f(x) is the position of the card in its column, after one dealing. Here the card positions range from 1 to 7.

x	f(x)
1	3
2	3
3	4
4	4
5	4
6	5
7	5

Note that f(f(x))=4 and this explains why the trick works.

Exercises / Questions for discussion:

1) (a) Modify the previous table where the card positions range from 0 to 6 instead of 1 to 7.
(b) Suppose now that the columns are numbered 0,1,2 in the order of dealing. Define using a table and a formula, a function g(x) having value the column number of the chosen card after dealing, where x is as above.
(c) There was no need to use g(x) to show that the trick works. Why?

2) Write a formula defining f(x) for the modified table.

3) Modify the original table where the card positions range from -3 to +3 instead of 1 to 7. What do you notice?

From now on suppose that:

$m,n \geq 1$ are integers.

The number of rows is odd $(2m+1) = (m+1+m)$.
This is also the length of a column.

The number of columns is odd $(2n+1) = (n+1+n)$.
This is also the length of a row.

Exercises / Questions for discussion:

1) Write a function for simulating a single dealing of the cards face down. Use Boolean values true, false to represent card vertical, card horizontal respectively. Write the function so that it works in the general case with (m+1+m) rows by (n+1+n) columns (and not just for 7 rows by 3 columns).

2) Write a sequential program and a parallel program which use the function you wrote in the previous question to simulate the 7 rows by 3 columns card trick with cards dealt face down. The sequential and parallel versions of the program should be similar. The trick should be simulated after the card is chosen. The trick should be simulated 7 times to check that it works with all possibilities.

3) Regarding the previous question, do you think that the parallel or sequential program runs faster?

The following table describes a function f(x) where x is the position of the card in the chosen column and f(x) is the position of the card in its column, after one dealing. The card positions range from -3 to +3 instead of 1 to 7.

x	f(x)
-3	-1
-2	-1
-1	0
0	0
+1	0
+2	+1
+3	+1

Movement property

A card not in the middle moves closer to the middle after dealing.
The middle card remains the middle card after dealing.

The previous function $f(x)$ satisfies:
$|f(x)| < |x|$ if $x \neq 0$;
$f(x)=0$ if $x=0$.

(There is a unique fixed-point where $f(x)=x$; i.e. $x=0$.)

The movement property holds more generally - explained later.

This explains why the trick works more generally if we deal the cards often enough.

So, if there are (m+1+m) rows, the question “In which column is the card you chose?” needs to be asked at most m times for the chosen card to reach the middle position of one of the columns. Therefore, this question needs to be asked one more time to identify the chosen column and card - all in all at most (m+1) times.

Observation / Lemma

After dealing, the cards from the chosen column will never appear in the first or last row.

Case (m+1+m) rows by 3 columns. This is because
# of cards in a column is $(m+1+m) \geq 3$ = # of cards in a row,
since $m \geq 1$.

Case (m+1+m) rows by (n+1+n) columns. Since n columns are put before and after the chosen column, the # of cards put before or after the chosen column is:
$(m+1+m)n \geq 3n \geq (n+1+n)$ = # of cards in a row,
since $m,n \geq 1$.

Generalization 1

Movement property: (m+1+m) rows by 3 columns.

Proof by induction on m:

Basis m=1; i.e. # of cards in a row = 3.
By our observation/lemma, the cards from the chosen column will never appear in the first or last row after dealing. As there are only three rows, they must therefore appear in the middle row; i.e. the movement property holds.

Alternatively, as there are 3 rows and 3 columns, deal 9 cards to check that the movement property holds in this case.

Prelude to the inductive step:

Combined columns <u>before</u> dealing

1 column of (m+1+m) cards

1 column – (m+1+m) cards – chosen column

1 column of (m+1+m) cards

Total: 3(m+1+m) cards.

Rows ***after*** *dealing*

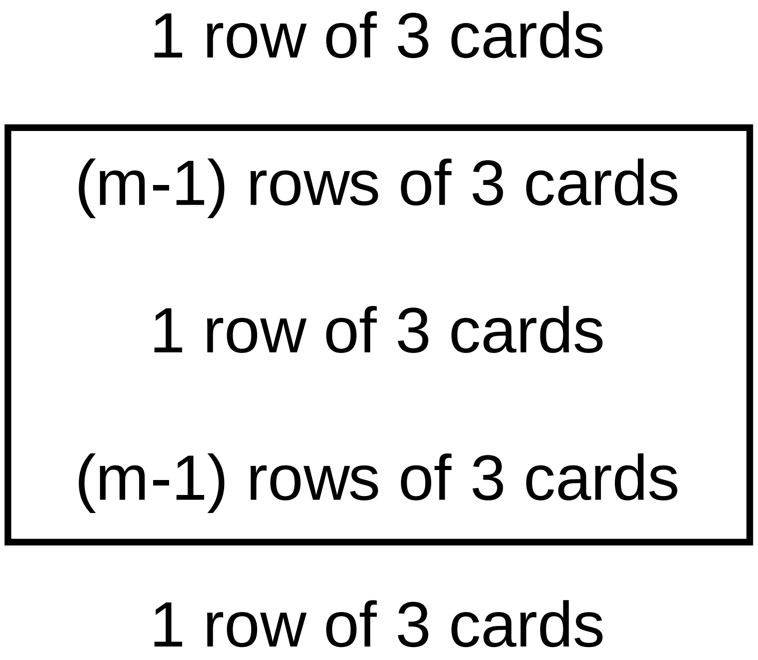
m rows of 3 cards

1 row of 3 cards

m rows of 3 cards

Total: (m+1+m)3 , cards as before.

Inductive step: By our observation, after dealing, the cards from the chosen column will never appear in the first or last row.



1 row of 3 cards

(m-1) rows of 3 cards

1 row of 3 cards

(m-1) rows of 3 cards

1 row of 3 cards

The box above represents the case (m-1) which works by inductive assumption.

Generalization 2

Movement property: (m+1+m) rows by (n+1+n) columns

Induction on m.

Basis m=1; i.e. # of cards in a row = (n+1+n).
By our observation, as there are only three rows, the cards from the chosen column must appear in the middle row after dealing.

Inductive step similarly works by our observation.
Diagrams are similar to those of Generalization 1.

Exercises / Questions for discussion:

1) Write a detailed proof of Generalization 2.

2) Explain from first principles why the trick works if row length (odd) is greater than or equal to the column length (odd). First experiment with the case row and column lengths equal.

3) In the previous question, how many times do we need to ask: In which column is the card you chose?
Do an experiment to understand this.

4) Suppose that the row length is odd (2n+1) and the column length is even (2m). Show that after dealing, the cards from the chosen column will never appear in the first or last row, when $m \geq 2$, $n \geq 1$.

This means that the chosen card will eventually end up in one of the two middle positions of the column chosen if the cards are dealt sufficiently often.

Comment: If the row length is even (2n), i.e. the number of columns is even, then there is no way of putting the chosen column in the middle of all the cards.

Part 3 - Doing prevents too much thinking

(My father’s second piece of advice, ע״ה)

Child in India: Wanted to do the trick 21 times to check that it works but then, play was the most important thing for me.

Youth in England: Formula for f(x) where the card positions range from 0 to 6.
f(x) = (x+7) DIV 3 where the value of DIV is the quotient of the division, e.g. (11 DIV 3) = 3.

Lecturer in Israel: Table for f(x) by numbering the cards in the column chosen from 1 to 7 and dealing the cards face down.

Pensioner in Israel: More on dealing the cards face down.
Parallelism for beginners. Formula g(x) = (x+7) MOD 3.

I thought too much!
I should have heeded both pieces of advice of my father!

My atonement: Used hands and head in the explanations.
Thought and action were combined.

Experiments in the exercises.

Summary

Multi-faceted example requiring problem solving as well as programming

Summarize what you see leaving out specific details.

Identify what is relevant.

Variety of viewpoints.

Use your imagination.

Experiment with the problem; not just program debugging.

Conclusion - What good is all this?

Guided activity.

First examples and exercises should have similar sequential and parallel solutions.

Preferably they should be solvable without a computer.

Parallelism should be introduced or demonstrated at the very beginning.

Flexible algorithms and execution.

Flexible Computation Research Laboratory
FLEXCOMP Lab

Educational material on Flexible algorithms and execution.

Embedded Flexible Language - EFL.

Flexible algorithms and EFL have an imperative style.

Hardware support for Flexible execution including a provisional patent.

Thank You

Web: flexcomp.jct.ac.il email: flexcomp@jct.ac.il
Flexible Computation Research Laboratory - FLEXCOMP Lab

Web: homedir.jct.ac.il/~rafi e-mail: rafi@g.jct.ac.il
Raphael B. Yehezkael