intel®
Education

Live Computing
Instructor's Handbook

Henryk Michalewski, Andrzej Nagórko

Education

Education

PREFACE

Authors Henryk Michalewski, Andrzej Nagórko

WARSAW
November 2013

Preface

Live Computing Project is an educational project. It is aimed at Middle and Secondary School students. Its goal is to introduce concepts of computer science to popular audience, without use of computers nor scientific terminology. Through a hand-on experience it lets students discover their talent for computational thinking.

Computational thinking is a collection of diverse skills to do with problem solving that result from studying the nature of computation. It includes some obviously important skills that most subjects help develop, like creativity, ability to explain and team work. It also consists of some very specific problem solving skills such as the ability to think logically, algorithmically and recursively. Computer Science is unique in the way it brings all these diverse skills together [?]. The spirit of the project resembles that of Computer Science Picnic, Computer Science Unplugged and of Science Festival.

This booklet contains scenarios of activities that were performed during the following three events

1. „Warsztaty My´slenia Problemowego", 18th May 2013, Department of Computer Science, University of Warsaw

2. „International Exchange: Żagle School (Warsaw, Poland) - Everest Academy (Lugano, Switzerland)", 18th, 19th, 20th November 2013, Szkoła Żagle.

3. „Live Computing", 22nd November 2013, Intel Gdańsk. It is intended for instructors wishing to make their own presentations.

How to use this booklet

The twenty activities described in this booklet can be composed in a variety of ways and are mostly independent. As a guideline we list the selection of activities that we did during the three events listed above. However, you are strongly encouraged to make your own selection, depending on the audience and the time available.

CONTENTS

Contents

Live Computing

Education

ACTIVITIES

3. ACTIVITIES

To teach effectively a teacher must develop a feeling for his subject; he cannot make his students sense its vitality if he does not sense it himself. He cannot share his enthusiasm when he has no enthusiasm to share. How he makes his point may be as important as the point he makes; he must personally feel it to be important.
George Pólya "Mathematical discovery" (New York, 1981)

This chapter contains description of the activities from the instructor point of view. Following the above G. Pólya quote, to perform those activities effectively you need to have a good background in the theory behind each activity. If you are not an expert in the field, reading the references provided in "Suggested Reading" section of each activity is strongly recommended. It is the responsibility of the instructor to tell the audience about the wide background of the activity. However, try to not give lectures, but react proactively to the discoveries that students make while performing themselves the exercises that you give.

3.1 Does Computer Think? The Turing Test

The Turing Test activity is a good opening activity if you plan to spend a lot of time with the student. It is designed to raise curiosity about how computer works. We tease audience with the question whether computer thinks and show examples of seemingly intelligent computer behaviours, but our point is quite the contrary: we stress that the computer does not think; all it does is a lot of simple calculations. What we see is the intelligence of the human programming the device. This activity is an invitation for the audience to a journey through the land of a computer science.

1. Ask the audience if computer thinks. Take a poll, write results on the blackboard. Usually, the audience will be unanimous that computer does not think: this would be a big news otherwise, wouldn't it? In the next minutes will try to convince them that the opposite is true.

2. Talk about examples of intelligent or human computer behaviour.

   (a) Computer chess. This is the traditional milestone of artificial intelligence: for years it was believed that it requires thought to play chess. The story of Kasparov - Deep Blue match shows otherwise (see [1]). The computers are now stronger than the chess masters.
   If the audience is interested, you can discuss the story of backgammon, where self-trained computer program surpassed the human masters (see [3]).

   (b) Creativity. Talk about computational creativity. [5]

   (c) Art. Computers can be talented artists as well: show fractal art. [4]

3. Take a poll again, write results on the blackboard. No one changed opinion?

4. Describe the Turing Test [6]

5. The students are shown four conversations (page 71):

   (a) Conversation #1

   (b) Conversation #2

   (c) Conversation #3

   (d) Conversation #4

Their task is to guess which side is a computer. Take a poll after showing each conversation, write the results on the blackboard. Make the audience that it can be computer-computer or human-human conversation as well, but don't make it too easy!

6. Reveal to the audience which were the computers.

7. Take another poll: Does computer think?

8. Fire up Eliza [2]. Take a volunteer, let him ask 8 questions to the computer. Again, the audience tries to guess if he talks with a human, or with a computer.

9. If the audience is interested, repeat this with another volunteer.

10. Tell the audience that they were talking with the Eliza program. Describe briefly how it works and its history.

11. This activity is meant to be the introductory one. It provides a good opening to the next activities: how does computer do it that it can do seemingly intelligent things? The point that we want to make is that what we see is intelligence, but the intelligence of programmers that did program the computer, not the computer itself.

3.1.1 Possible Extensions

You may want to do the Turing test with the A.L.I.C.E. chatbot, the current state-of-the-art.

3.1.2 The Eliza program

You'll find an adapted Eliza chatbot along with the usage instructions at the following address http://livecomputing.org/turingtest

3.1.3 Suggested Reading

[1] World chess champion Garry Kasparov:

A week after the match, Kasparov expressed his admiration for Deep Blue's play in game 2: The decisive game of the match was Game 2, which left a scar in my memory and prevented me from achieving my usual total concentration in the following games. In Deep Blue's Game 2 we saw something that went well beyond our wildest expectations of how well a computer would be able to foresee the long-term positional consequences of its decisions. The machine refused to move to a position that had a decisive short-term advantage – showing a very human sense of danger. I think this moment could mark a revolution in computer science that could earn IBM and the Deep Blue team a Nobel Prize. Even today, weeks later, no other chess-playing program in the world has been able to evaluate correctly the consequences of Deep Blue's position. "

J. Schaeffer, A. Plaat: "Kasparov versus Deep Blue: The Re-match", http://webdocs.cs.ualberta.ca/~jonathan/PREVIOUS/Grad/Papers/db.html

[2] Eliza Chatbot Implementation, http://livecomputing.org/chatbot.

[3] G. Tesuaro: "Programming backgammon using self-teaching neural nets", Artificial Intelligence 134 (2002) 181-189.

[4] "ICM 2006 Benoit Mandelbrot Fractal Art Contest", http://www.fractalartcontests.com/2006/

[5] Wikipedia contributors, "Computational Creativity", Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/Computational_creativity (accessed November 2013)

[6] G. Oppy, D. Dowe: "The Turing Test", The Stanford Encyclopedia of Philosophy (Spring 2011 Edition), Edward N. Zalta (ed.), http://plato.stanford.edu/archives/spr2011/entries/turing-test/

[7] "Transcripts of 2013 Loebner Competition", http://www.appsentience.com/loebner2013/Loebner2013.htm

3.1.4 What you'll need

Eliza chatbot, blackboard, slides with conversations.

3.1.5 Age

The activity is suitable for ages 13+.

3.2 Counting in Binary

This activity is Computer Science Unplugged "Binary Numbers" activity and we include it here for completeness. Refer to [8] for details.

3.2.1 Suggested Reading

[8] CS Unplugged: "Count the Dots - Binary Numbers", http://csunplugged.org/sites/ default/files/activity_pdfs_full/unplugged-01-binary_numbers.pdf

3.2.2 What you'll need

Dot-cards, partially burnt CD. Involves 6 volunteers. Takes up to 30 minutes.

3.2.3 Age

The activity is suitable for ages 9+.

3.3 The Adding Machine

The goal of this activity is to show how binary addition works with a mechanical calculator - the Adding Machine. The activity gives a good explanation what a processor clock is and what it means for a processor to be 32- or 64-bit (incidentally, our mechanical calculator is a 6-bit machine). This activity should be preceded by the *Counting in Binary* activity.

1. Start with the presentation of how the Adding Machine works. See [9] for a guide.

2. Using an analogy with how the Adding Machine works explain the following topics.

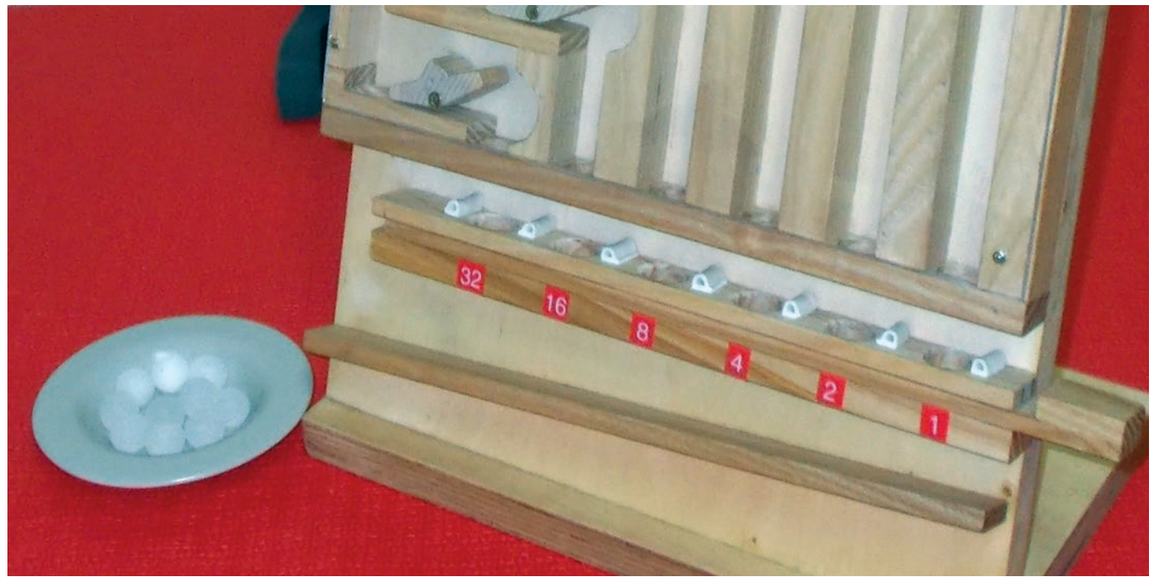


Figure 3.1: The Adding Machine

   (a) Computer clock. The marbles are the electrical signals. The frequency of how often you put the marbles in is frequency of the clock. To help students visualize what 1 GHz is talk about the nanostick [10].

   (b) Explain how the Adding Machine is a 6-bit computer.

3.3.1 Suggested Reading

[9] Matthias Wandel: "Marble Adding Machine", http://www.youtube.com/watch?v=GcDshWmhF4A

[10] Grace Hopper: "Admiral Grace Hopper explains the Nanosecond", http://www.youtube.com/watch?v=9eyFDBPk4Yw

3.3.2 What you'll need

The Adding Machine. The activity takes up to 30 minutes.

3.3.3 Age

The activity is suitable for ages 9+.

3.4 Image Representation

This activity is based on Computer Science Unplugged "Fax Machine" activity. It explains what a pixel is and explains the RLE (Run Length Encoding) compression algorithm. See [11] for more.

3.4.1 Suggested Reading

[11] CS Unplugged: "Colour by Numbers - Image Representation", http://csunplugged.org/sites/default/files/activity_pdfs_full/unplugged-02-image_representation.pdf

3.4.2 Age

The activity is suitable for ages 7+.

3.5 Error Correcting Codes

This activity builds on Computer Science Unplugged "Magic Card Flip" [?].

1. Pick one volunteer and perform the Magic Card Flip [12].

2. Can the audience guess how the trick was done? Do not tell them, let them guess! Repeat the trick if necessary.

3. Divide audience into pairs (or fours) and let them sit together. Let them practice the magic card flip: one student takes role of the instructor, the other flips the card. Use small two-sided squares that accompany this booklet for this (the students work on a table or on a carpet - the squares are not magnetic).

4. Once the students understand the trick give them worksheet #1 and discuss if it is possible to detect two errors with the code and if it is possible to correct two errors with the code.

5. Ask groups that finished early to design a code that will allow to correct more than one error (a possible hint: use more than two directions, e.g. diagonals).

3.5.1 Extra Credit Assignments

International Standard Book Number (ISBN) error-correcting code.

3.5.2 Suggested Reading

[12] CS Unplugged: "Magic Card Flip",

3.5.3 What you'll need

Big magnetic squares, metal blackboard, small squares (single set for each team), worksheet #1.

Figure 3.2: Students working on the error correcting codes

3.5.4 Age

The activity is suitable for ages 10+. Extra credit assignments 15+.

3.6 Text Representation - Huffman Encoding

This activity shows how computer can represent text data with just 0'1 and 1's. We start by introducing the Morse code and through the Huffman codes we discuss an efficient text encoding scheme.

3.6.1 The Activity

The activity requires slides from the web page http://livecomputing.org/slides
Follow the slides to explain the Morse code and the Huffman encoding to the students. The students will solve worksheet #2 in this activity.

3.6.2 What you'll need

Slides, worksheet #2.

3.6.3 Age

The activity is suitable for ages 15+.

3.7 Ciphers

This activity shows the benefits of treating characters as numbers. The ASCII table is introduced and the students learn about the Caesar cipher.

3.7.1 The Activity

The activity requires slides from the web page http://livecomputing.org/slides
Follow the slides to explain the ASCII table and the Caesar cipher to the students. The students will solve worksheet #3 in this activity. The worksheet #4 is an extra assigment that you can give students to solve at home.

3.7.2 Suggested Reading

[13] C. Savarese, B. Hart: "The Caesar Cipher", http://www.cs.trincoll.edu/~crypto/historical/caesar.html

3.7.3 What you'll need

Slides, worksheets #3 and #4, cipher wheels.

3.7.4 Age

The activity is suitable for ages 12+.

3.8 Sorting: why do we care?

Theare two goals to this activity: the first one is to introduce the divide and conquer algorithms; the second one is to emphasize the importance of sorting in computer science.

3.8.1 The Activity

1. Perform the birthday cake presentation, see [14] for details.

2. Ask for a volunteer. Give him a shuffled deck of cards, with 10 of spades switched for 8 of hearts (from another deck). Ask him to check if there are all cards in the deck (he should do it in his place while you go on with the next step).

3. Perform the binary search presentation, see [15] for details. Explain the divide and conquer strategy.

4. Ask the volunteer if there are all cards in the deck. How did he found out? Point the error if he just counted cards and give the deck to the next person. To detect that a card is missing people will usually order cards: point this fact out.

5. Show the audience the "missing letter" slide. Ask them to find out the missing letter. Show the ordered letters and explain how ordering helps us find the missing one.

3.8.2 Suggested Reading

[14] CS Unplugged: "Birthday cake", http://www.youtube.com/watch?v=TDid6XMUm10

[15] CS Unplugged: "Binary search", http://csunplugged.org/searching-algorithms

[16] Live Computing: "A missing letter", http://livecomputing.org/slides.

3.8.3 What you'll need

A cake, knife and candles. A prepared deck of cards. Cups, ping-pong balls with numbers. Slides (the slides are available on http://livecomputing.org/slides). Involves 4 volunteers. Suggested time: up to 40 minutes.

3.8.4 Age

The activity is suitable for ages 12+.

3.9 Stolen Treasure

The goal of this activity is to solve a classical fake coin problem (and its variants) using the Divide and Conquer strategy.

3.9.1 The Activity

This activity involves a classical fake coin problem: you have a set of coins. One coin is fake: it is lighter than the rest. You have a balance scale and have to find out which coin is fake. In our variant we have treasure chest and one chest is missing part of the treasure. Hint for instructor: these chests are marked by a red dot underneath: use it to set up the problem for students.

Figure 3.3: The stolen treasure activity.

1. Present the problem to the audience and solve the fake coin problem for three chests. (You can solve it with just one weighting).

2. Pick a volunteer. Let him solve the problem for four chests. How many weightings do you need? (Now you need to use the scale twice.)

3. Solve problem for 7 chests. Count number of weightings aloud. Can anyone do better? (You can do it in two weightings.)

4. Ask the audience how many weightings do you need for 9 chests. (Two are still enough.) Ask a volunteer to solve a problem with just two weightings.

5. Consider how big a problem you can solve with three weightings (up to 27 chests). Talk the audience through a variants of the problem: what if you don't know if the fake tresure is lighter or heavier than the rest? What if you have two fake chests? See [17] for more.

3.9.2 Extra Credit Assignments

Two extra credit assignments that you can do if the audience is interested ([18], [19]).

1. Among 10 given coins, some may be real and some may be fake. All real coins weigh the same. All fake coins weigh the same, but have a different weight than real coins. Can you prove or disprove that all ten coins weigh the same in three weighings on a balance scale?

2. Among 100 given coins, four are fake. All real coins weigh the same. All fake coins weigh the same, but they are lighter than real coins. Can you find at least one real coin in two weighings on a balance scale?

3.9.3 Suggested Reading

[17] M. Kołodziejczyk: "Waga szalkowa i uogólniony problem fałszywej monety" (in Polish), http://math.uni.lodz.pl/~andkom/Marcel/Kule.pdf

[18] J. Soprunova: "Fake Coins and a Balance Scale", MATH 64091, Kent State University, Spring 2010. http://www.math.kent.edu/~soprunova/64091/weight10.pdf

[19] T. Khovanova: "Scary Coins", Tanya Khovanova's Math Blog, http://blog.tanyakhovanova.com/?p=248

3.9.4 What you'll need

One or two balance scales, treasure chests. Involves 1-2 volunteers. The activity takes up to 1 hour.

3.9.5 Age

The activity is suitable for ages 10+.

3.10 L-tromino puzzle

In this activity the students fill a square with L-trominoes and one monomino, discovering and applying the Divide and Conquer principle to solve the problem. The problem is divided into four smaller subproblems, each solved by a recursive application of the strategy.

3.10.1 The Activity

See [21] for a description of the problem and its solution.

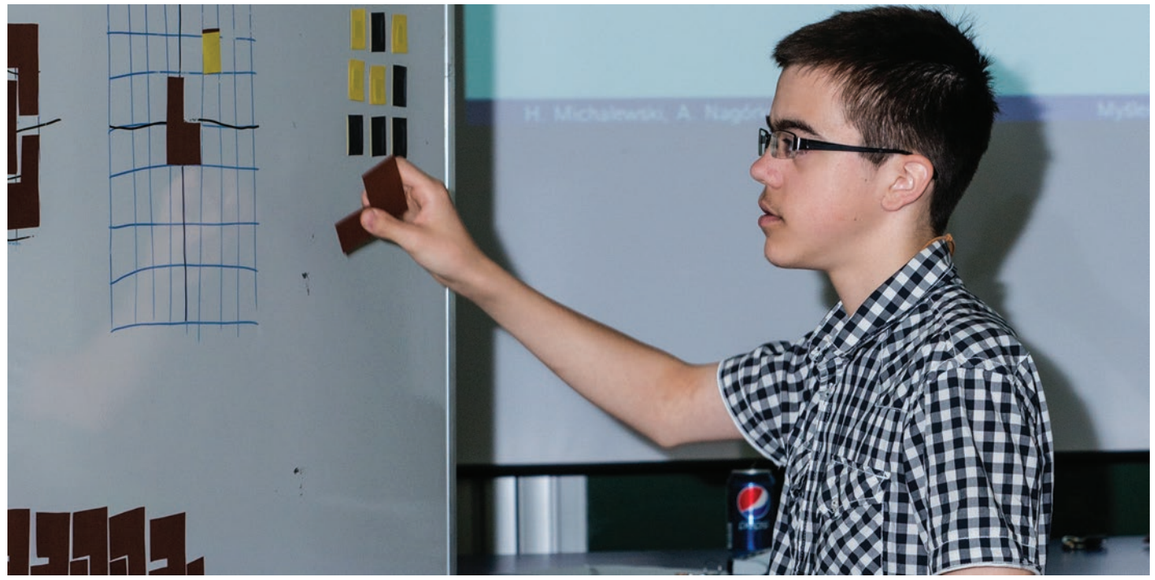


Figure 3.4: A student solving the L-tromino puzzle.

1. Prepare a blackboard with 22, 44 and 88 grids drawn, one monomino and enough L-shaped trominoes.

2. Pick a volunteer to solve a 22 puzzle. This is a very easy and instant task: use it to activate the audience (don't forget the candy!).

3. Pick a volunteer to solve a 44 puzzle. Let audience choose the place where the monomino is placed. Guide the student if he is stuck.

4. Try to find a recursive strategy with the audience.

5. Pick a volunteer to solve a 88 puzzle.

6. Discuss how you use divide and conquer strategy to solve the problem. Discuss how to problem gets easier when we divide it and reduce it to something that we already know how to solve.

3.10.2 Suggested Reading

[20] S. W. Golomb: "Polyominoes", Princeton University Press, 1994, page 5.

[21] N. Starr: "The Tromino Puzzle", http://www3.amherst.edu/~nstarr/trom/intro.html

3.10.3 What you'll need

One monomino, 27 L-shaped trominoes, metal blackboard, chalk. Involves 3 volunteers. Takes up to 20 minutes.

3.10.4 Age

The activity is suitable for ages 10+.

3.11 Sorting Network

In this activity students learn about sorting networks and design comparison networks of their own. The first part of this activity is based on [22].

1. Do the sorting network activity, as described in [22].
2. Split the group into teams of up to four students.
3. Give worksheet #5 to the students. Explain the problem on the blackboard. Let the students solve it on their own. Check each group's solution, point out the the errors if there are any.
4. Give worksheets #6-#11 to the students one by one. For each worksheet pick a student that will present his team's solution on the blackboard.
5. Students quite often get stuck with the larger networks. Give hints. Example of an hint: to find the second largest element you can find the largest one, discard it and find the largest one from the remaining elements.
6. Challenge the good students to find networks that contain:
   (a) the least number of comparators,
   (b) the least number of levels.
7. Discuss the analogy between the Adding Machine and a sorting network. See [23] and [24] for more.

3.11.1 Suggested Reading

[22] CS Unplugged: "Sorting Networks", http://csunplugged.org/sorting-networks.

[23] D. Knuth: "Sorting and Searching", The Art of Computer Programming, Vol. 3, 2nd ed., section 5.3.4.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: "Introduction to Algorithms, 2/e", chapter 27. https://mitpress.mit.edu/sites/default/files/Chapter\%2027.pdf

3.11.2 What you'll need

Sorting network, cards with numbers, worksheets, blackboard. The activity takes up to 1 hour.

3.11.3 Age

The activity is suitable for ages 10+.

2.12 Bubble Sort

The goal of this activity is to teach students a bubble sort algorithm and familiarize them with the sorting paradigm. Some students, when faced with a sorting problem, will do many ad-hoc optimizations, e.g. on a sorting network student with the largest number often will decide to go directly to his spot, bypassing comparators and not obeying comparison rules. The setup of this activity is such that no such shortcuts are possible.

3.12.1 The Activity

1. Pick cards (ten cards of same color, e.g. spades, suffice). Shuffle, put on table face down, in a row.
2. Pick a volunteer. Explain the rules to the volunteer. The rules are:
   (a) You (the instructor) will never look at the cards directly. You depend on the volunteer to compare them.
   (b) You'll show cards to the volunteer in pairs - holding left card from the row in your left hand and right card from the row in your right hand.
   (c) When you show cards the volunteer has to tell you "switch" if the left card (from his perspective) is bigger than the right card or "no" if the left card is smaller.
3. After you make sure that the first vounteer understands the rules, pick another volunteer, who will count the number of comparisions.
4. Perform the bubble search on the cards (see [25] for detailed description of the algorithm):
   (a) Pick up the first and the second card. If the volunteer says "switch", put the cards in reversed order; otherwise put them back without changing the order.
   (b) Do the same with second and third card. Switch or not switch - depending on what volunteer says.
   (c) Proceed to the next pair, until the last one. The second volunteer counts (aloud) the number of comparisons (i.e. the number of times that you show the cards).
   (d) Repeat the process, until all cards are sorted.
   (e) Write the number of comparisons on the blackboard.
5. It is unlikely, but it might happen that there was not much to do after shuffling. In this case comment on it and repeat the sorting with another volunteer.
6. Explain to the audience the bubble sort algorithm, explain from what the name comes from.
7. Divide the group into groups of three and let them perform the sort in groups. One students takes your role (the most responsible one - he has to understand how the bubble sort works), another two students take roles of first and second volunteer. The students work with a worksheet #4.12 (Exercise 1 - bubble sort).

3.12.2 Suggested Reading

[25] E. Grimson, J. Guttag: "Introduction to Computer Science and Programming. Lecture 9: Binary search, bubble and selection sorts". MIT Open Courseware. http: //ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/video-lectures/lecture-9/

3.12.3 What you'll need

The presentation requires a deck of cards and involves two volunteers. The work in groups requires a set of cards for each group of three students. The activity takes 20-60 minutes.

3.12.4 Age

The activity is suitable for ages 12+.

3.13 Selection Sort

The goal of this activity is to teach students selection sort and familiarize them with the sorting paradigm. It is similar to the "Bubble Sort" activity - only the algorithm explained differs.

Activities

Live Computing

3.13.1 The Activity

Follow the steps from "Bubble Sort" activity, but use selection sort (see [26]) instead of bubble sort.

3.13.2 Suggested Reading

[26] E. Grimson, J. Guttag: "Introduction to Computer Science and Programming. Lecture 9: Binary search, bubble and selection sorts". MIT Open Courseware. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/video-lectures/lecture-9/

3.13.3 What you'll need

The presentation requires a deck of cards and involves two volunteers. The work in groups requires a set of cards for each group of three students. The activity takes 20-60 minutes.

3.13.4 Age

The activity is suitable for ages 12+.

3.14 Quick Sort

The goal of this activity is to teach students quick sort and familiarize them with the sorting paradigm. It is similar to the "Bubble Sort" activity - only the algorithm explained differs.

3.14.1 The Activity

Follow the steps from "Bubble Sort" activity, but use quick sort (see [27]) instead of bubble sort. Remind the students about the Birthday Cake example and Binary Search and remind them how halving the problem allows us to solve it quicker. Show how quick sort (usually) halves the problem it has to solve. Compare with selection sort.

3.14.2 Suggested Reading

[27] MIT OpenCourseWare: "1.204 Computer Algorithms in Systems Engineering", Lecture 9, http://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec09.pdf

3.14.3 What you'll need

The presentation requires a deck of cards and involves two volunteers. The work in groups requires a set of cards for each group of three students. The activity takes 20-60 minutes.

3.14.4 Age

The activity is suitable for ages 12+.

3.15 Sorting Dances

The goal of this activity is to compare running times of bubble sort, selection sort and quick sort. This may replace or be replaced by the "Sorting with a Scale" activity. It is better suited for a larger group of students.

Figure 3.5: Quick-sort with Hungarian folk dance.

3.15.1 The Activity

1. Split the audience into three equal groups. Give each group a same set of cards with numbers.

2. Ask each group to stand in a row in such a way that numbers are in the same order (but are not ordered).

3. Each group will perform one the sorting algorithms: bubble, selection or quick sort.

4. A single step of our "dance" is as follows: a pair of students from each group comes forward (the pair is determined by the rules of a particular algorithm - see "Bubble Sort", "Selection Sort" and "Quick Sort" activities for details). They compare their numbers and switch or not, depending on the result of the comparison.

5. Keep everything synchronized, i.e. a next comparison is made only after all three groups are done with the previous move.

6. Count the number of comparisons. Write results on the blackboard.

7. Which algorithm was the quickest one? I highly recommend to run [28], [29] and [28] movies in the background during the activity, with sound on!

3.15.2 Suggested Reading

[28] AlgoRythmics: "Quick-sort with Hungarian (Küküll˝omenti legényes) folk dance", http://youtu.be/ywWBy6J5gz8

[29] AlgoRythmics: "Select-sort with Gypsy folk dance", http://www.youtube.com/watch?v=Ns4TPTC8whw

[30] AlgoRythmics: "Bubble-sort with Hungarian ("Csángó") folk dance", http://youtu.be/IyZQPjUT5B4

3.15.3 What you'll need

Cards with numbers (three sets, one for each type of sorting algorithm). Optionally - a projector. The activity takes 20-40 minutes.

3.15.4 Age

The activity is suitable for ages 10+.

3.16 Sorting with a Scale

The goal of this activity is same as the goal of "Sorting Dances" activity: to compare running times of bubble, selection and quick sort algorithms and talk about computational complexity. It is better suited for smaller group of students.

3.16.1 The Activity

Students get a set of different weights in similarily packaged boxes that are indistinguishable to the eye. The goal is to order weights in order from lightest to heaviest. The student is allowed to compare only two weights at once, using a scale. He may reorder weights making decisions based on those measurements only. The instructor keeps track of how many weightings were taken.

1. Give weights to the first student. Let him sort the weights using bubble sort algorithm. Count the number of comparisons. Write the result on the blackboard.

2. Give weights to the second student. Let him sort the weights using selection sort algorithm. Count the number of comparisons. Write the result on the blackboard.

3. Give weights to the first student. Let him sort the weights using quick sort algorithm. Count the number of comparisons. Write the result on the blackboard.

4. Remind the students about the Birthday Cake example and Binary Search and remind them how halving the problem allows us to solve it quicker. Show how quick sort (usually) halves the problem it has to solve. Compare with selection sort.

Please note that the number of weightings (comparisons) depends on the (random) order of weights that are given. It might happen that the bubble sort will beat the quick sort. This is good opportunity to discuss difference between complexity of a problem and complexity of an individual instance of a problem. If this ever happen, discuss the difference and run the experiment again!

3.16.2 What you'll need

A scale, weights. Blackboard. Three volunteers. Takes 20-60 minutes.

3.16.3 Age

The activity is suitable for ages 12+.

3.17 Tower of Hanoi and Exponential Growth

The goal of this activity is to familiarize students with an example of a task that requires exponentially many operations to solve, no matter how smart you are. Two solutions are shown: a recursive and an iterative one that illustrate differences between the two approaches. The exponential lower bound on the complexity of the problem is proven.

3.17.1 The Activity

1. Explain the "Tower of Hanoi" problem (see [31]) using the supplied materials.

2. Take a volunteer, let him solve problem for 3 disks. This is very easy, do it quickly and make everyone laugh. Don't forget to give a candy to the volunteer!

3. Take a volunteer, let him solve problem for 4 disks. Point audience towards the fact that it took much longer than the last try.

4. Take a volunteer, let him solve problem for 5 disks. Point audience towards the fact that it took much longer than the last try.

5. Take a volunteer, let him solve problem for 6 disks. This is optional, as the solution will take minimally 63 steps. Do it only if audience seems interested.

6. Commend volunteers for their work and their thinking. Now we'll solve the problem without thinking! Explain the iterative algorithm (see [32]).

7. Take two volunteers. One will perform the iterative algorithm, the other will count the number of operations performed. Your task is to ensure that the volunteer performing the algorithm does it correctly and smoothly and to write the number of opeartions on the blackboard.

8. The first volunteer solves problem for 2 and 3 disks. The other volunteer counts opeartions. Note on the blackboard the number of opeartions in the following format

Disks	2	3		
Operations	3	7		

9. Change the volunteer that performs the iterative algorithm. Let him solve it for 4 disks. Update your table:

Disks	2	3	4	
Operations	3	7	15	

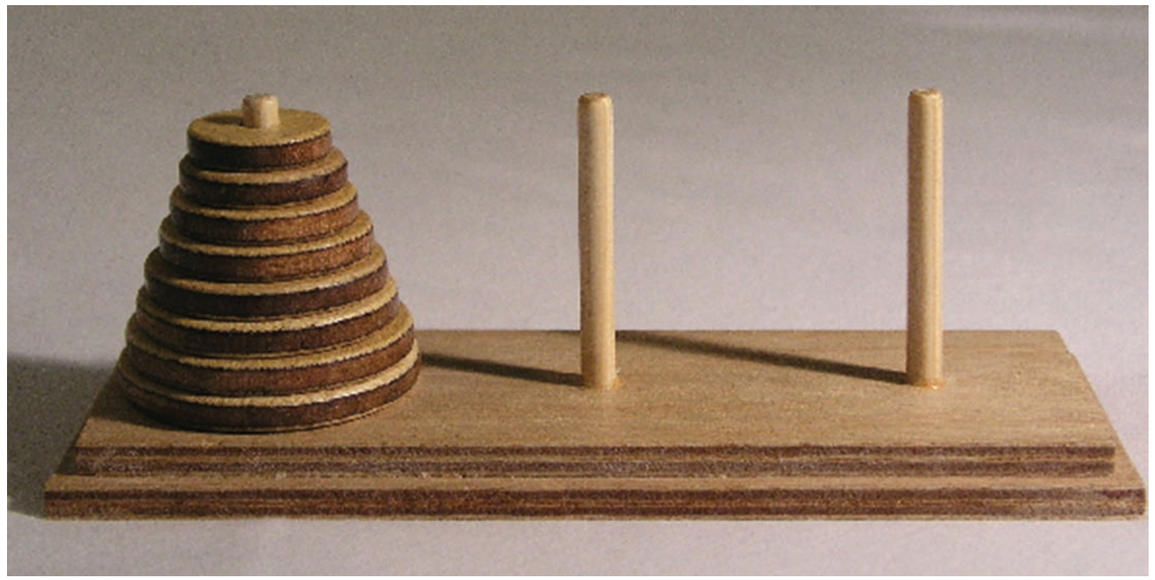


Figure 3.6: The Tower of Hanoi.

10. Ask audience if they see a pattern. If they do, confirm the pattern doing the next task. If they don't, give them more data doing the next task.

11. Change the volunteer that performs the iterative algorithm. Let him solve it for 5 disks. Update your table:

Disks	2	3	4	5	
Operations	3	7	15	31	

12. Discuss the pattern with audience.

13. Optionally, solve the problem for 6 disks (if the audience is really interested). Everyone can count aloud.

14. Dicuss the recursive version of the procedure and (informally) show the formula for the required number of steps:

    $S(n+1) = 2\ S(n)+1$:

15. Verify that it fits with the observation for small n's that you checked with iterative algorithm. Comment that it proves that the iterative algorithm was optimal in the cases that you checked. Inform that this is true in general.

16. Talk about exponential growth and how large the number 2^n is for n equal to 10, 20, 50, 100. Read [35] for ideas.

17. If you performed the Birthday Cake activity earlier, talk about connection between the two.

3.17.2 Suggested Reading

[31] S. Devadas, E. Lehman: "Recurrences", in "6.042/18.062J Mathematics for Computer Science. March 17, 2005 Lecture Notes" http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-spring-2005/lecture-notes/l12_recur2.pdf

[32] Wm. R. Franklin: "A simpler iterative solution to the Towers of Hanoi problem". ACM SIGPLAN Notices 01/1984; 19(8):87-88. http://www.ecse.rpi.edu/~wrf/p/28-sigplan84-hanoi.pdf

3.17.3 What you'll need

Towers of Hanoi, blackboard. Involves 2 - 8 volunteers. The activity takes 30-50 minutes (depending mostly on your choice if you solve the puzzle for 6 disks or not).

3.17.4 Age

The activity is suitable for ages 12+.

3.18 Travelling Salesman Problem and Computational Complexity

In this activity students compete to find a best solution to an optimization problem. Travelling Salesman Problem (TSP) is a classical problem that represents a vast class of NP-complete problems, the problems that we don't know better algorithms than the exponential ones. See [33] for a background.

The problem has real-life applications in processor design [34] and serves as a good introduction to talk about P vs NP conjecture and its consequences [36], the value of approximate solutions and about limits of computation [38]. Students explore various instances of the problem, competing to find the best solution. Then we compare the solutions with the ocmputer-generated ones and discuss how complexity grows with increasing number of cities. We discuss computational complexity [35], P versus NP conjecture [36] and the Millenium Problems [39].

3.18.1 The Setup

The activity is done using what we call TSP Boards. These are wooden boards with drilled holes in a regular grid. Students put pegs into the holes to represent cities and construct the route that connects the cities using a thread. One can measure the total length of the route by measuring how much thread is left. The problems are distributed on worksheets with picture of the grid of the cities. Once the students decide on their best solution, they draw it on the worksheet and submit for evaluation. We let teams of 4 persons work in team with one TSP board.

This activity can be also carried out outdoors with pegs and a piece of rope. We recommend this although it takes a lot more time to do this this way.

3.18.2 The Activity

1. Explain the Travelling Salesman Problem to the audience (see [33]), using the large TSP board. A good map to introduce the problem is the 16 cities Ulysses route from the Odyssey.

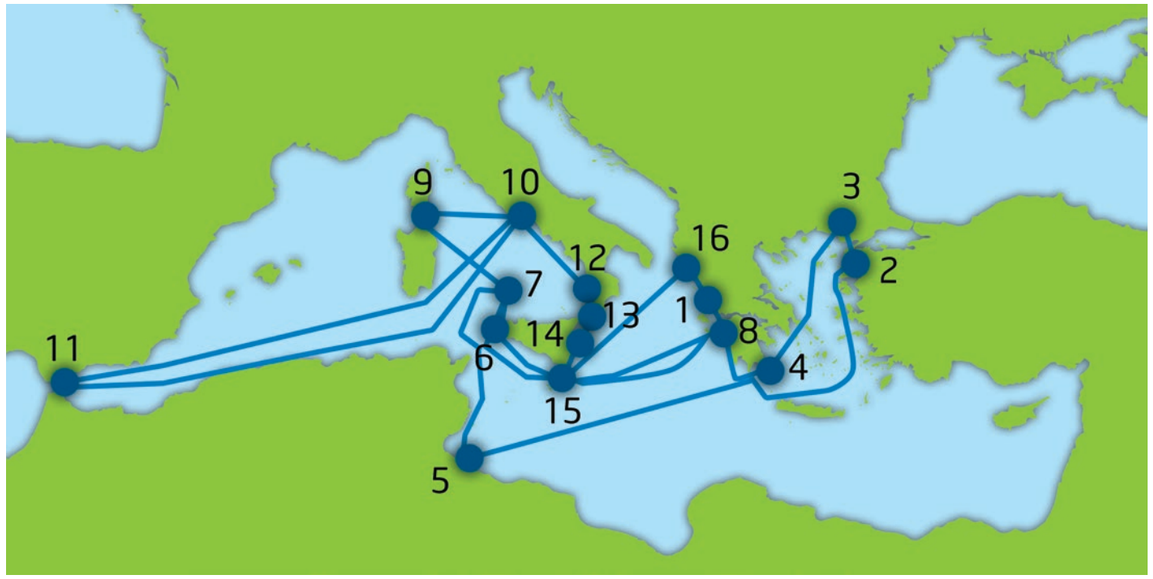


Figure 3.7: The Ulysses route from the Odyssey [33].

2. Proceed with worksheets (#1 - #4). Collect results. Announce to the whole class records each time the record is broken (let the audience applaud the best team!)

3. After the time is up, let the team with the best result show it to others. Show the audience the optimal computer-generated solution (either from slide, on the large TSP board or with printouts).

4. Discuss the complexity of a naive algorithm that would check all possible route (it is factorial, explain why on small examples). Use scientific calculator to observe, that the number grows faster than the Tower of Hanoi number (i.e. faster than 2n). [citecomplexity

5. Discuss the P versus NP problem, Millennium prizes and why the problem is important. [36]

6. Discuss the success stories of record-breaking algorithms that solve TSP instances with thousands of cities. How is it possible? [33] Discuss the difference between solving single instance of an NP-hard problem over solving a complete class.

3.18.3 Suggested Reading

[33] M. Grötschel, M. Padberg: "The Optimized Odyssey", AIRONews VI, n.2 Summer 2001, https://www.zib.de/groetschel/pubnew/paper/groetschelpadberg2001a.pdf

[34] "VLSI Data Sets" http://www.math.uwaterloo.ca/tsp/vlsi/

[35] M. James: "Computational Complexity", i-programmer.info http://www.i-programmer. info/babbages-bag/499-computational-complexity.html

[36] L. Fortnow: "The Golden Ticket: P, NP, and the Search for the Impossible", pp. 192, Princeton University Press 2013.

[37] S. Aaronson: "NP-complete Problems and Physical Reality", http://www.scottaaronson.com/papers/npcomplete.pdf

[38] D. Harel: "Algorithmics: The Spirit of Computing", pp. 536, 3rd Edition, Addison-Wesley 2004.

[39] S. Cook: "The P versus NP Problem", Clay Institute, http://www.claymath.org/sites/default/files/pvsnp.pdf

3.18.4 What you'll need

TSP Boards (one small board for each team, large board for yourself), pegs, threads, tapemeasure, projector (if you want to display optimal solutions), blackboard (if you want to keep track of the best scores). Involves whole group (divided into teams). The activity takes 1-3 hours, depending on how many optimization problems you give. The outdoor activity takes much longer, as only one solution can be checked at a time. You will need help of another person if you work with teams, to be able to measure solutions concurrently.

3.18.5 Age

The activity is suitable for ages 12+.

3.19 Steiner Problem

This activity introduces another NP-hard optimization problem to the students: the Steiner problem. The goal is to find a network of routes between cities that minimizes the total distance of the roads built. It has a beautiful geometric background. See [41] for more and [42] for interesting historical background.

3.19.1 The Setup

The TSP Boards used in the Travelling Salesmen Problem will be helpful to explore the problem, although it can be done with pen and paper only.

3.19.2 The Activity

In the activity we solve the Steiner Problem in two particular instances: a triangle of cities and a rectangle of cities. The understanding of a solution of a first instance helps students to discover solution of the second one on their own.

1. Explain the Steiner problem to the students and discuss (on the blackboard) the case for a triangle. Start with equilateral triangle. Measure the length of the tree if we take two edges as the network. Measure the length of the tree if we build the tree through the middle point of the triangle. With older and more mathematically inclined audience you can discuss the general case. Follow [40] for details.

2. Divide students into teams and let them work on the rectangular Steiner problem, using worksheet #17. They can experiment using the supplied TSP Boards. Help the students measure their results and announce to everyone when new record is set.

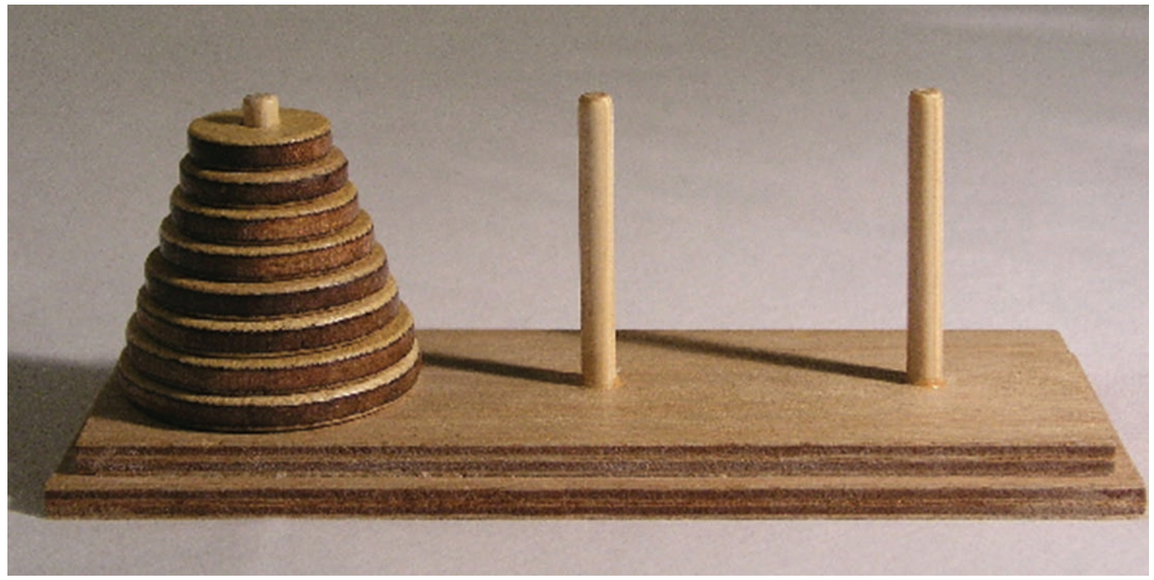
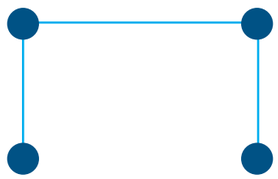
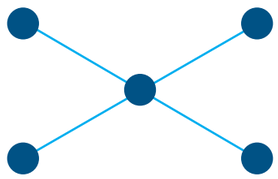


Figure 3.6: The Tower of Hanoi.

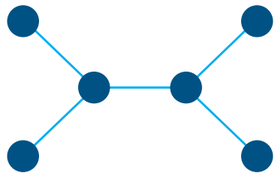
3. Here is a natural progression of ideas:



We can do better going through the mid point:



Now we see two triangles and we know that it is better to go through the Steiner points in each of the triangles:



There is no better solution, i.e. you cannot find a shorter network for rectangle if you allow more than 2 Steiner points.

3.19.3 A possible extension

You can use soap bubbles to "solve" the Steiner problem. See [43] and [41] for details.

3.19.4 Suggested Reading

[40] J. Willis: "The Shortest Network Connecting Three Points", http://www.uvm.edu/~jmwillis/public_html/CH7_45.pdf

[41] D. Thomas: "Target? TARGET? We don't need no stinkin' Target!", pandasthumb.org http://pandasthumb.org/archives/2006/07/target-target-w-1.html

[42] X. Cheng, Y. Li, D.-Z. Du, H. Q. Ngo: "Steiner Trees in Industry", Handbook of Combinatorial Optimization (Vol 5), pp. 193-216. http://cs.gsu.edu/~yli/papers/steiner.pdf

Before 1967, the charge of a private [phone] network was determined by the length of the minimum spanning tree for the destinations. The minimum spanning tree for a set of terminals is the shortest tree with edges betweenterminals. It is different from the Steiner tree

by disallowing the existence of Steiner points. This restriction causes the minimum spanning tree possibly larger than the Steiner minimum tree for the same set of terminals. In 1967, a flight company found this fact. Therefore, they requested some new services at those Steiner points, so that the minimum spanning tree for the new set of destinations is the Steiner minimum tree for the original set of destinations, which is shorter than the minimum spanning tree of the original set of destinations. Therefore, those requests increased the service of telephone company and decreased the charge from the telephone company. With this situation, the telephone company had to change the billing base from minimum spanning tree to the Steiner minimum tree.

[43] S. Aaronson: "What Google won't find", a talk at Google Cambridge http://www.scottaaronson.com/blog/?p=266.

If quantum computers can't solve NP-complete problems in polynomial time, it raises an extremely interesting question: is there any physical means to solve NP-complete problems in polynomial time?

Well, there have been lots of proposals. One of my favorites involves taking two glass plates with pegs between them, and dipping the resulting contraption into tub of soapy water. The idea is that the soap bubbles that form between the pegs should trace out the minimum Steiner tree — that is, the minimum total length of line segments connecting the pegs, where the segments can meet at points other than the pegs themselves. Now, this is known to be an NP-hard optimization problem. So, it looks like Nature is solving NP-hard problems in polynomial time!

3.19.5 What you'll need

TSP Boards, worksheets, tape measure, blackboard (to keep track of scores and to discuss the optimal solutions). The activity is for whole group (divided into teams). It takes 30-60 minutes.

3.19.6 Age

The activity is suitable for ages 12+.

intel® Education

EVENTS

4. EVENTS

4.1 Pilot presentation

18th May 2013, Department of Computer Science, University of Warsaw

An open presentation held at the Department of Computer Science, University of Warsaw aimed at Middle School students. Its goal was to show what a talent for computational thinking is. Some of the interaction with the students required material displayed on slides. The slides are available from a web page http://livecomputing.org/slides/. The presentation was in Polish language.

During the presentation we talked about four aspects of computational thinking: problem division and recursion, grouping, patterns, abstraction and algorithmic thinking. We used Sierpi´nski carpet, L-trominoes, Binary Search, Sorting and Error correcting codes activities. The presentation was 2 hours long.

4.2 International Exchange

18th - 20th November 2013, Z˙agle School, Warsaw

A series of presentations for an international group of boys from the ˙ Zagle School (Warsaw, Poland) and Everest Academy (Lugano, Switzerland) that was an educational part of the international exchange between the school. It was attended by 10 boys from ˙ Zagle and 9 boys from Everest. They were split into 5 mixed teams beforehand and all worksheet activities were done in teams. The presentation was in English language.

We used candy to encourage interaction (it was given for every answer given by the audience), which proven to be very effective. We encourage you to do the same!

4.2.1 The Lectures

1. **"Does Computer Think?"**
   An opening activity with a goal to raise curiosity about how computer works. Based on the "Turing Test" activity.

2. **"What you can do with 0's and 1's?"**
   A set of activities that explain how computer represents data. Based on "The Adding Machine", "Image Representation", "Error Correcting Codes", "Text Representation - Huffman Encoding", "Ciphers" activities.

3. **"Divide and Conquer."**
   A set of activities that show a smart way to process data - divide and conquer algorithms and recursion. Based on "Sorting: why do we care?", "Stolen Treasure", "L-tromino puzzle" activities.

4. **"Sorting."**
   Based on "Sorting: why do we care?", "Sorting Network", "Bubble Sort", "Quick Sort", "Sorting Network" activities.

5. **"Computational Complexity".**
   Based on "Tower of Hanoi and Exponential Growth", "Travelling Salesman Problem and Computational Complexity" and "Steiner Problem" activities.

Education

WORKSHEETS

5. WORKSHEETS

This chapter includes worksheets that accompany the activities listed. The list of all included worksheets:

#1 Error correcting codes.

#2 Huffman encoding.

#3 Caesar cipher.

#4 Caesar cipher: home challenge.

#5 Sorting network (4 elements).

#6 Sorting network (5 elements).

#7 Sorting network (8 elements).

#8 Sorting network (second element out from four).

#9 Sorting network (second element out from eight).

#10 Sorting network (third element out from eight).

#11 Sorting network (smallest and largest).

#12 Sorting complexity.

#13 Travelling Salesman Problem. Small map.

#14 Travelling Salesman Problem. Medium map.

#15 Travelling Salesman Problem. Big map.

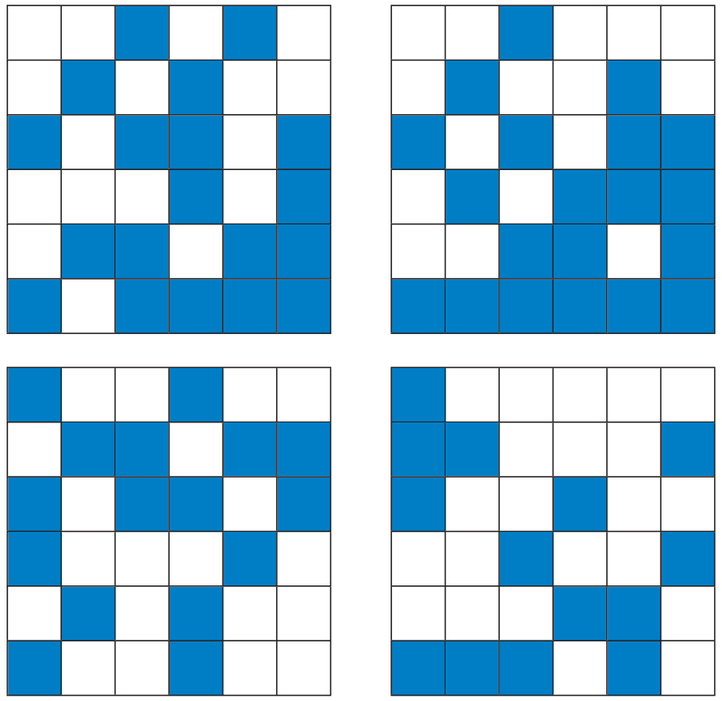
#16 Travelling Salesman Problem. Ultimate map.

#17 Steiner Problem.

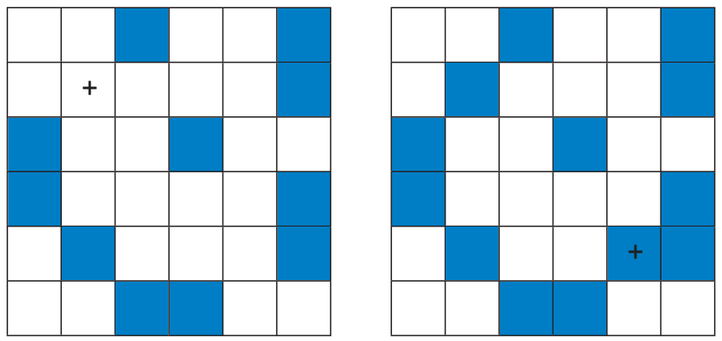
5.1 Worksheet #1. Error correcting codes.

Names: . . . . . .

Exercise. Find all flipped cards and mark them with crosses. No more than 2 cards are flipped.



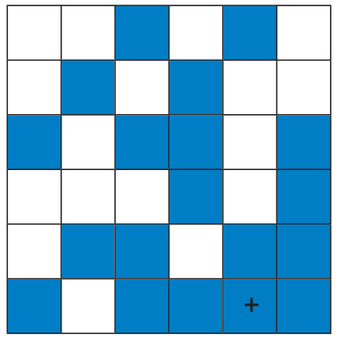
Example. Find all flipped cards and mark them with crosses. No more than 2 cards are flipped.



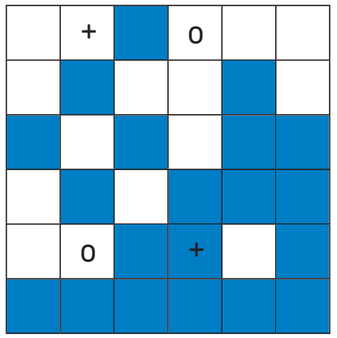
Worksheet #1. Error correcting codes. Solutions.

Names: . . . . . .

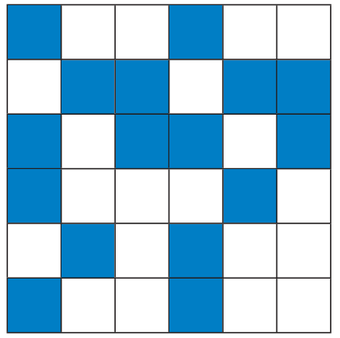
Exercise. Find all flipped cards and mark them with crosses. No more than 2 cards are flipped.



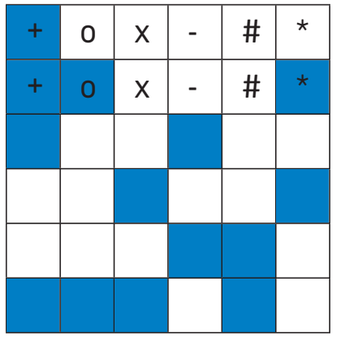
A single card is flipped.



Two possible solutions.

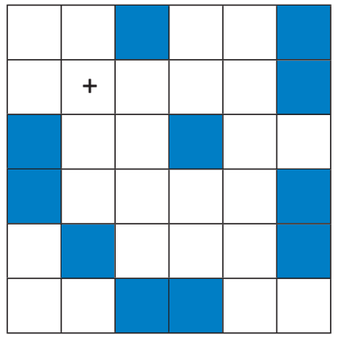


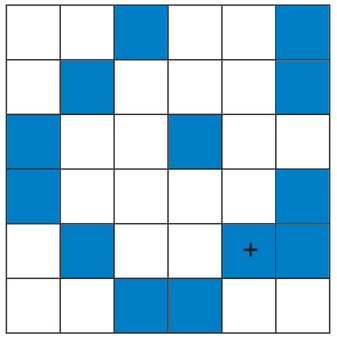
No cards are flipped.



Six possible solutions.

Example. Find all flipped cards and mark them with crosses. No more than 2 cards are flipped.





5.2 Worksheet #2. Huffman encoding.

Names: . . . . . .

Exercise. You've got the following text (obviously some sort of a cipher!):

BAGDEFCABHIBIG

Pick Huffman codes for every letter that you think will get shortest encoding for the above sentence.

A		D		G	
B		E		H	
C		F		I	

Hint: count how many times each letter occurs in the text.

Example. Example. For a text

ACBABA

the best encoding is

A	(3 occurences)	.
B	(2 occurences)	- .
C	(1 occurence)	- -

It encodes the text as . - - - . . - . . (9 bits).

Worksheets

Live Computing

4.2 Worksheet #2. Huffman encoding.

Names: . . . . . .

Exercise. You've got the following text (obviously some sort of a cipher!):

BAGDEFCABHIBIG

Pick Huffman codes for every letter that you think will get shortest encoding for the above sentence.

A	..-	D	.--.	G	.-.
B	-.	E	--.-	H	--..
C	.---	F	---	I	...

Hint: count how many times each letter occurs in the text.

Example. Example. For a text

ACBABA

the best encoding is

A	(3 occurences)	.
B	(2 occurences)	- .
C	(1 occurence)	- -

It encodes the text as . - - - . . - . . (9 bits).

5.3 Worksheet #3. Caesar cipher.

Names: .....

Exercise. Decipher the following message encoded with the Caesar cipher.

FRJLWRHUJRVXP

Your solution:

.....

Worksheet #3. Caesar cipher. Solution

Names: .....

Exercise. Decipher the following message encoded with the Caesar cipher.

FRJLWRHUJRVXP

Your solution:

.....COGITOERGOSUM.....

5.4 Worksheet #4. Caesar cipher: home challenge.

Names: .....

Exercise. Decipher the following message encoded with the extended Caesar cipher.

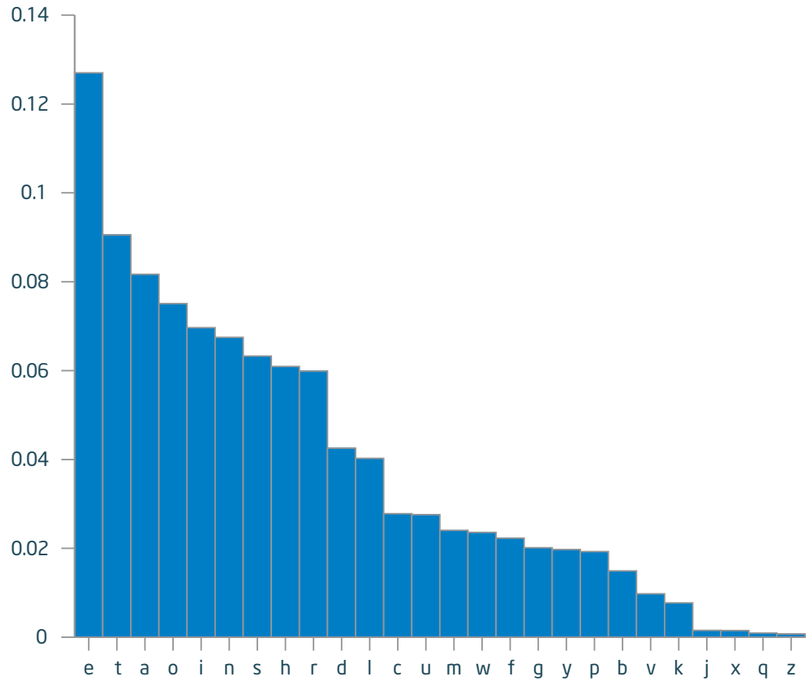
TFDGLKVIJTZVETVZJEFDFIVRSFLKTFDGLKVIJ
KYRERJKIFEFDPZJRSFLKKVCVJTFGVJ

Your solution:

.....

.....

Hint. Frequency of letters in the English alphabet.



Worksheet #4. Caesar cipher: home challenge. Solution

Names: .....

Exercise. Decipher the following message encoded with the extended Caesar cipher.

TFDGLKVIJTZVETVZJEFDFIVRSFLKTFDGLKVIJ
KYRERJKIFEFDPZJRSFLKKVCVJTFGVJ

Your solution:

.....COMPUTERSCIENCEISNOMOREABOUTCOMPUTERS.....

.....THANASTRONOMYISABOUTTELESCOPES.....

The alphabet is shifted by 17 places. The cipher is a famous quote by famous Dutch computer scientist Edsger Dijkstra.

5.5 Worksheet #5. Sorting network (4 elements).

Names: . . . . . .

Exercise. Design a sorting network that picks a largest number out of 4 (four) elements. Try to use as little comparators as you can. Draw your design below:

5.6 Worksheet #6. Sorting network (5 elements).

Names: . . . . . .

Exercise. Design a sorting network that picks a largest number out of 5 (five) elements. Try to use as little comparators as you can. Draw your design below:

5.7 Worksheet #7. Sorting network (8 elements).

Names: . . . . . .

Exercise. Design a sorting network that picks a largest number out of 8 (eight) elements. Try to use as little comparators as you can. Draw your design below:

5.8 Worksheet #8. Sorting network (second element out from four).

Names: . . . . . .

Exercise. Design a sorting network that picks a second largest number out of 4 (four) elements. Try to use as little comparators as you can. Draw your design below:

5.9 Worksheet #9. Sorting network (second element out from eight).

Names: . . . . . .

Exercise. Design Design a sorting network that picks a second largest number out of 8 (eight) elements. Try to use as little comparators as you can. Draw your design below:

5.10 Worksheet #10. Sorting network (third element out from eight).

Names: . . . . . .

Exercise. Design a sorting network that picks a third largest number out of 8 (eight) elements. Try to use as little comparators as you can. Draw your design below:

5.11 Worksheet #11. Sorting network (smallest and largest).

Names: . . . . . .

Exercise. Design a sorting network that picks a smallest and largest number out of 8 (eight) elements. Try to use as little comparators as you can. Draw your design below:

5.12 Worksheet #12. Sorting complexity.

Names: . . . . . .

Exercise 1. Count how many times you have to compare two cards when you sort the following sequence

10 5 4 7 8 9 6

using *bubble* sort.Your answer:

. . . . . .

Exercise 2. Count how many times you have to compare two cards when you sort the following sequence

10 5 4 7 8 9 6

using selection sort. Your answer:

. . . . . .

Exercise 3. Count how many times you have to compare two cards when you sort the following sequence

10 5 4 7 8 9 6

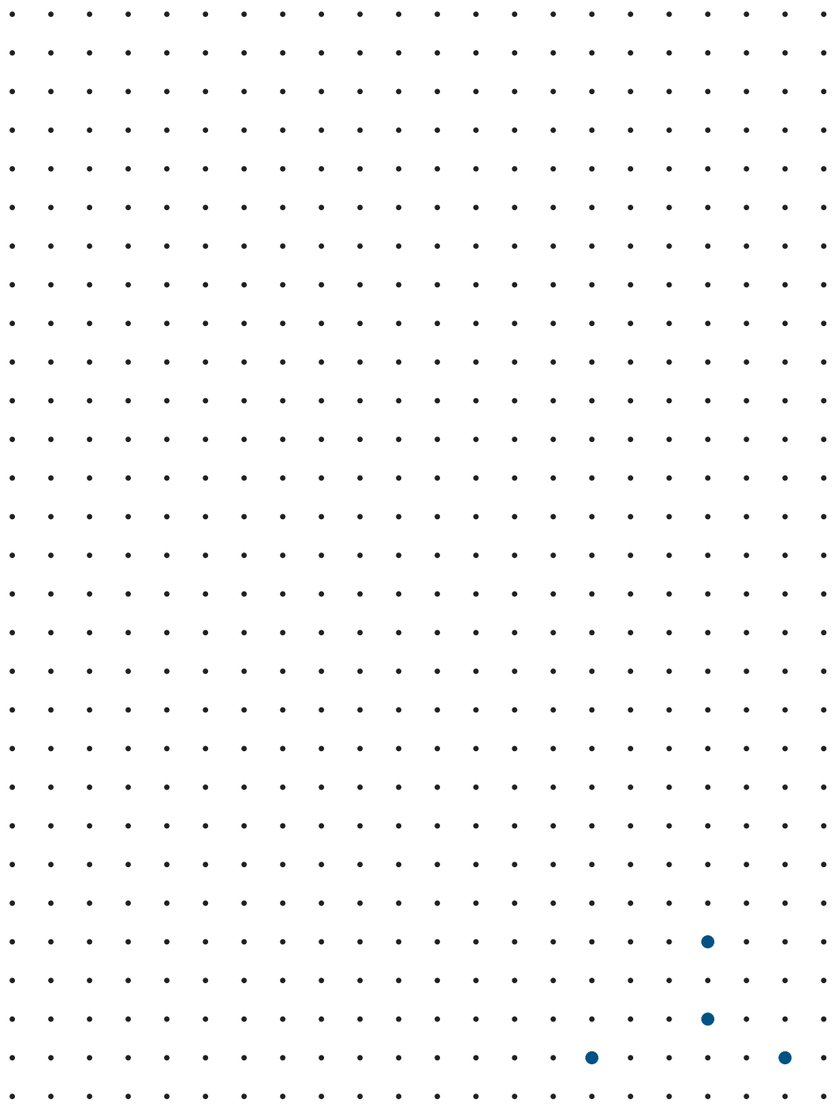
using quick sort.Your answer:

. . . . . .

5.13 Worksheet #13. Travelling Salesman Problem. Small map.

Names: .....

Exercise 1. Draw the best route that you found and note your result below.

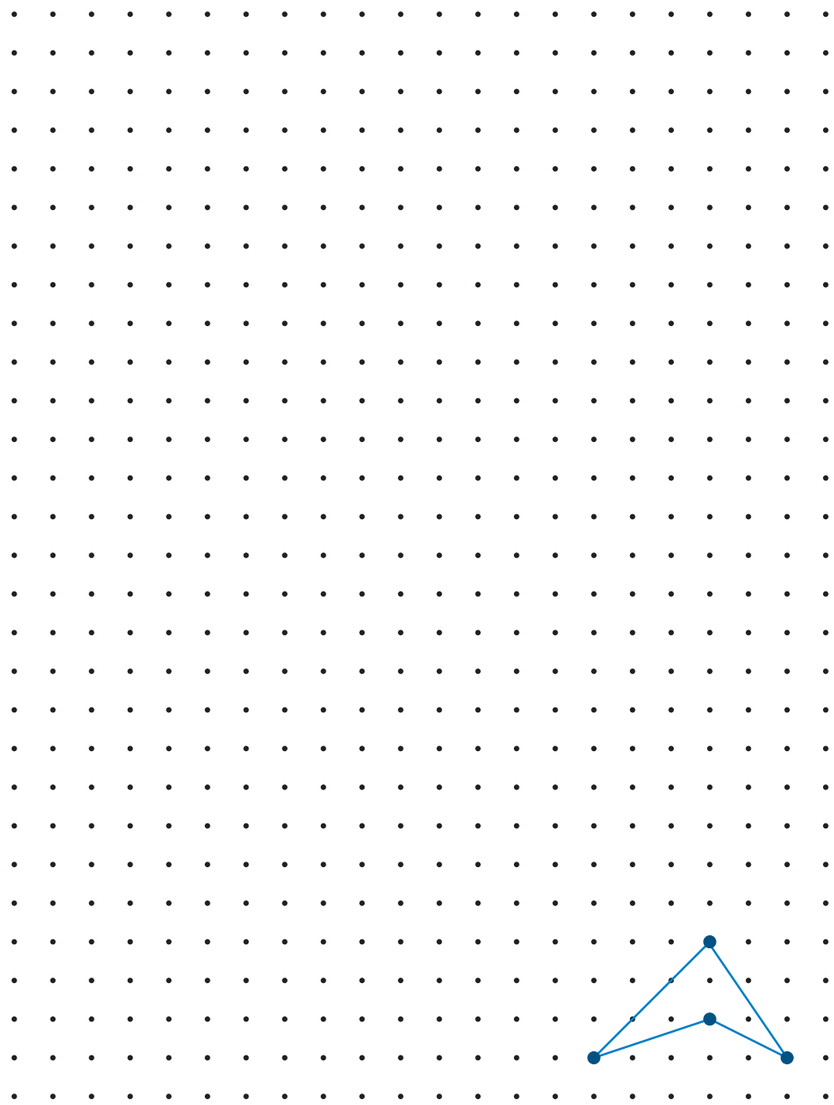


Best result: .....

Worksheet #13. TSP Solution. Small map.

Names: .....

Exercise 1. Draw the best route that you found and note your result below.

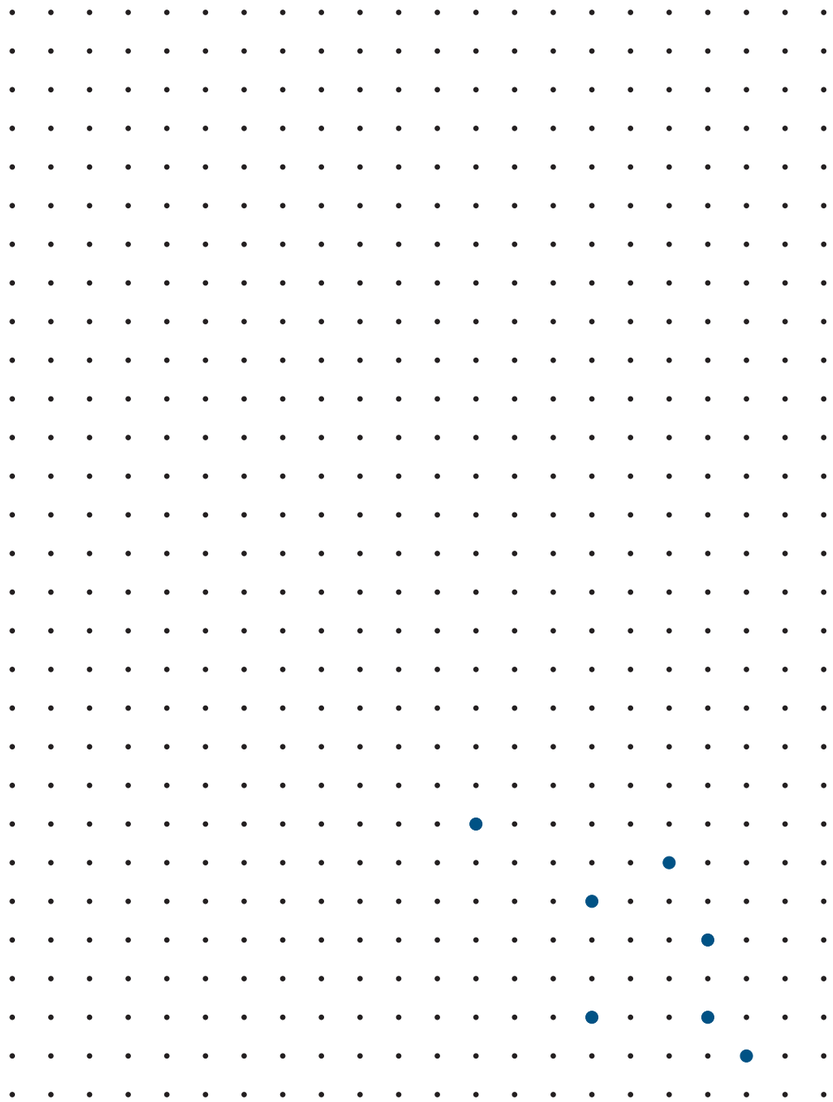


Best result: .....

5.14 Worksheet #14. Travelling Salesman Problem. Medium map.

Names: .....

Exercise 1. Draw the best route that you found and note your result below.

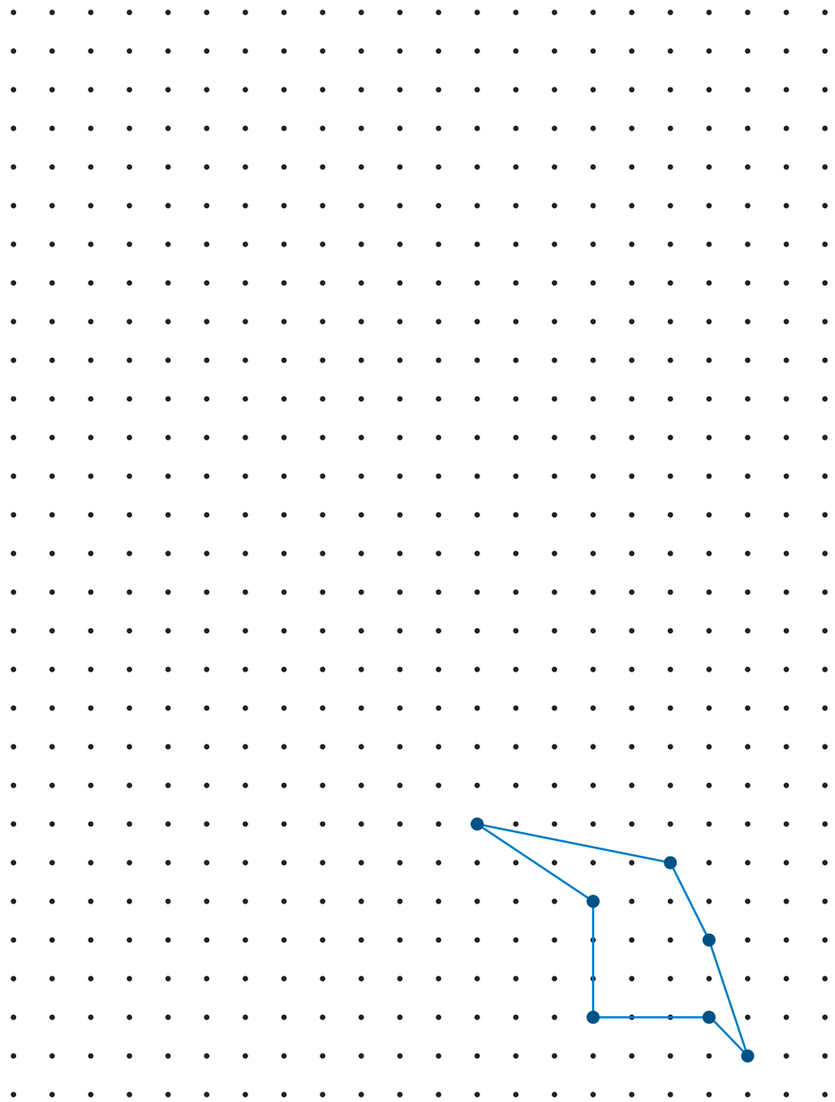


Best result: .....

Worksheet #14. TSP Solution. Medium map.

Names: .....

Exercise 1. Draw the best route that you found and note your result below.

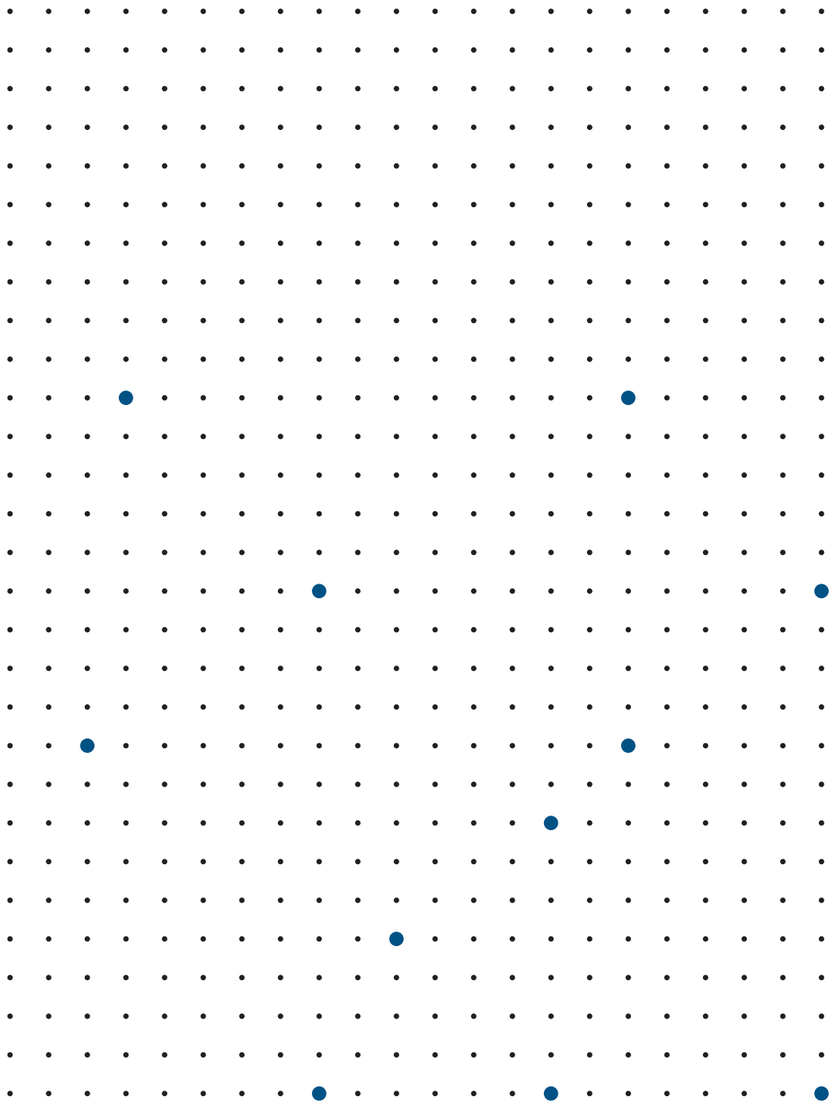


Best result: .....

5.15 Worksheet #15. Travelling Salesman Problem. Big map.

Names: . . . . . .

Exercise 1. Draw the best route that you found and note your result below.

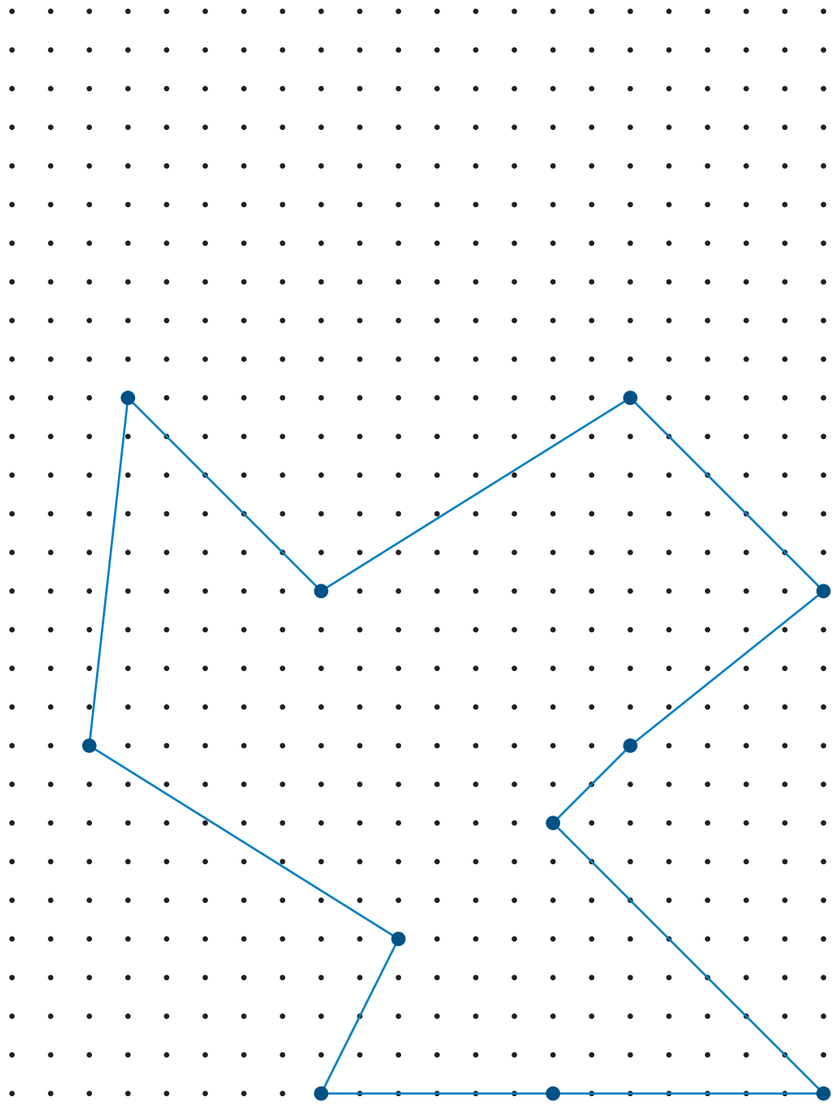


Best result: . . . . . .

Worksheet #15. TSP Solution. Big map.

Names: . . . . . .

Exercise 1. Draw the best route that you found and note your result below.

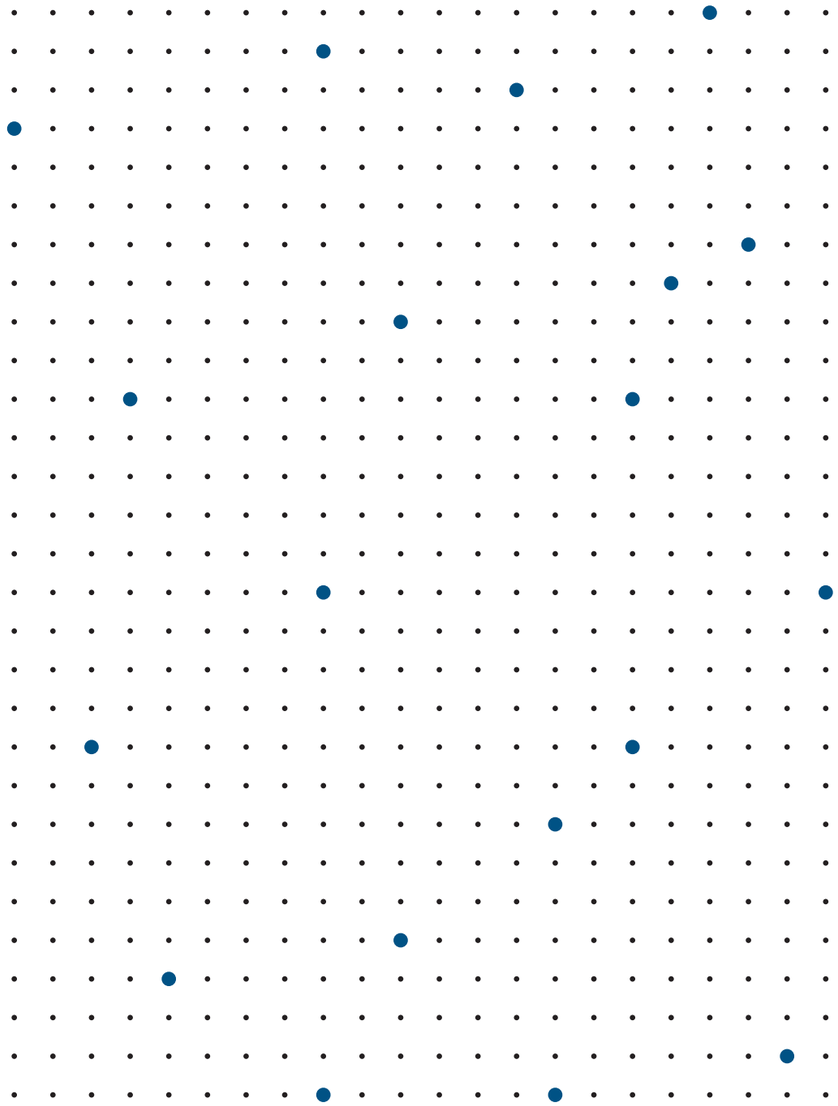


Best result: . . . . . .

5.16 Worksheet #16. Travelling Salesmen Problem. Ultimate Map.

Names: . . . . . .

Exercise 1. Draw the best route that you found and note your result below.



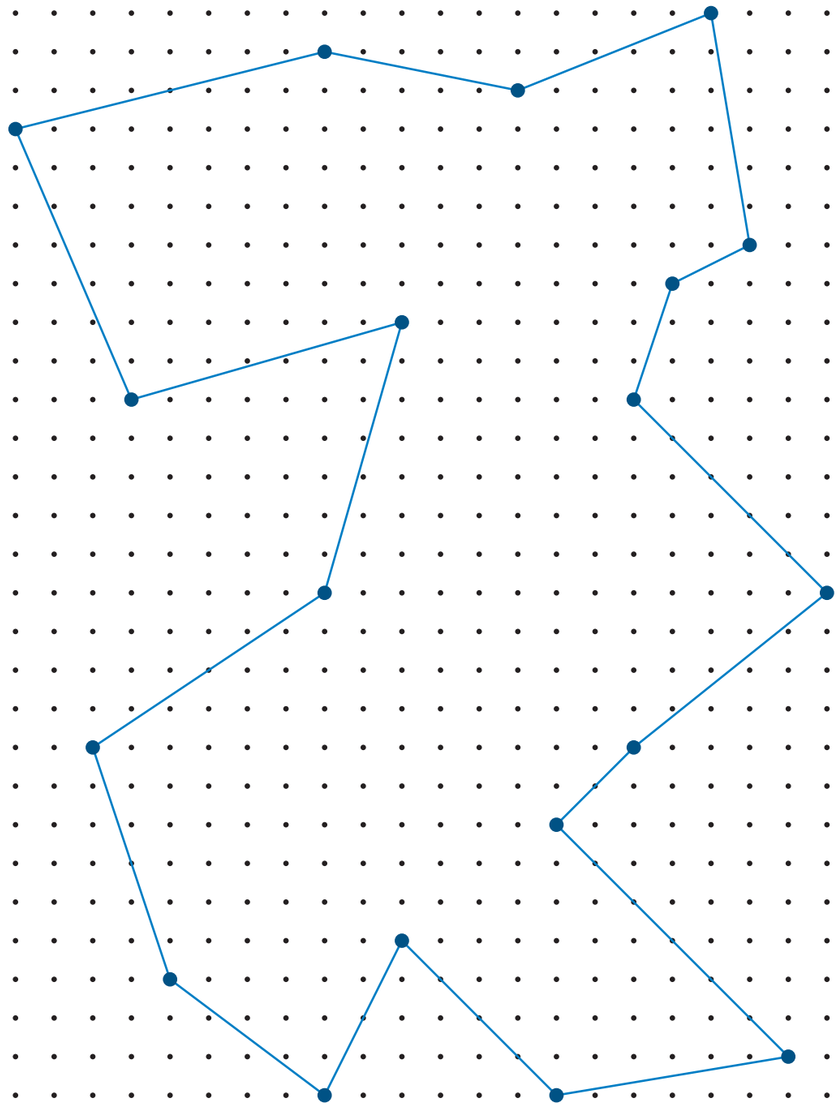
Best result: . . . . . .

Worksheets

Worksheet #16. TSP Solution. Ultimate map.

Names: . . . . . .

Exercise 1. Draw the best route that you found and note your result below.

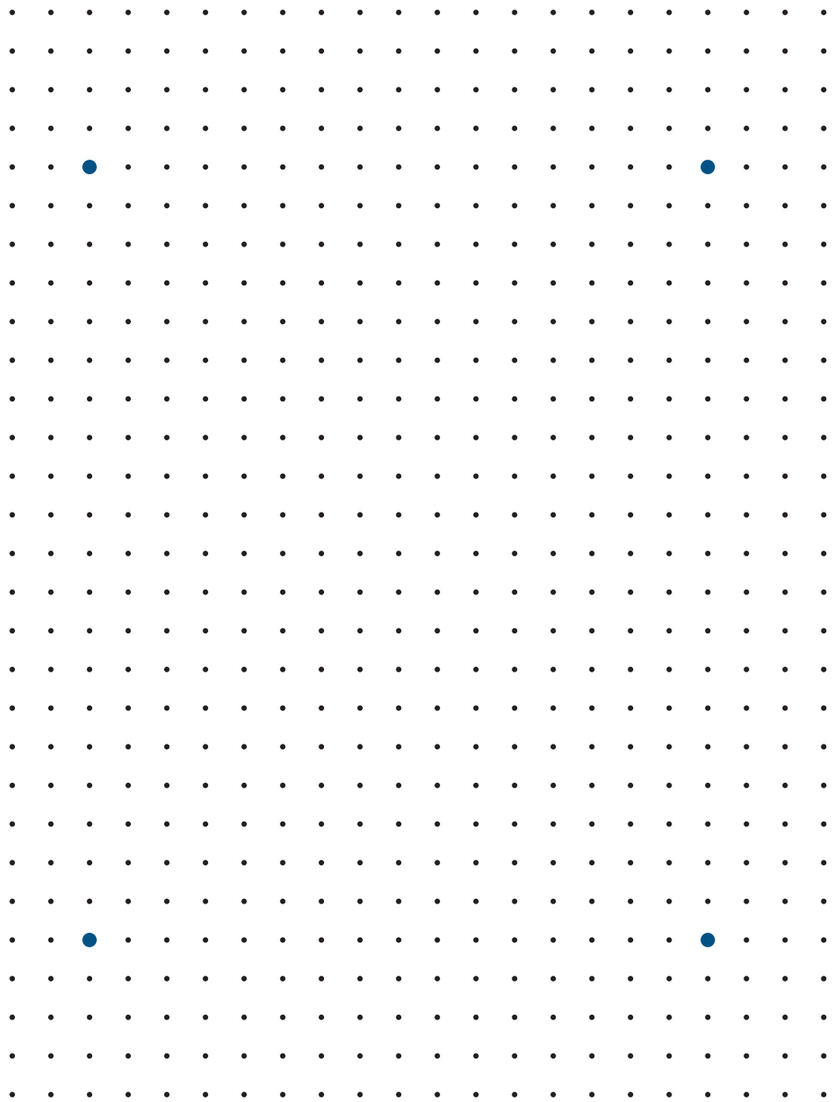


Best result: . . . . . .

5.17 Worksheet #17. Steiner Problem.

Names: . . . . . .

Exercise 1. Draw the best route that you found and note your result below.

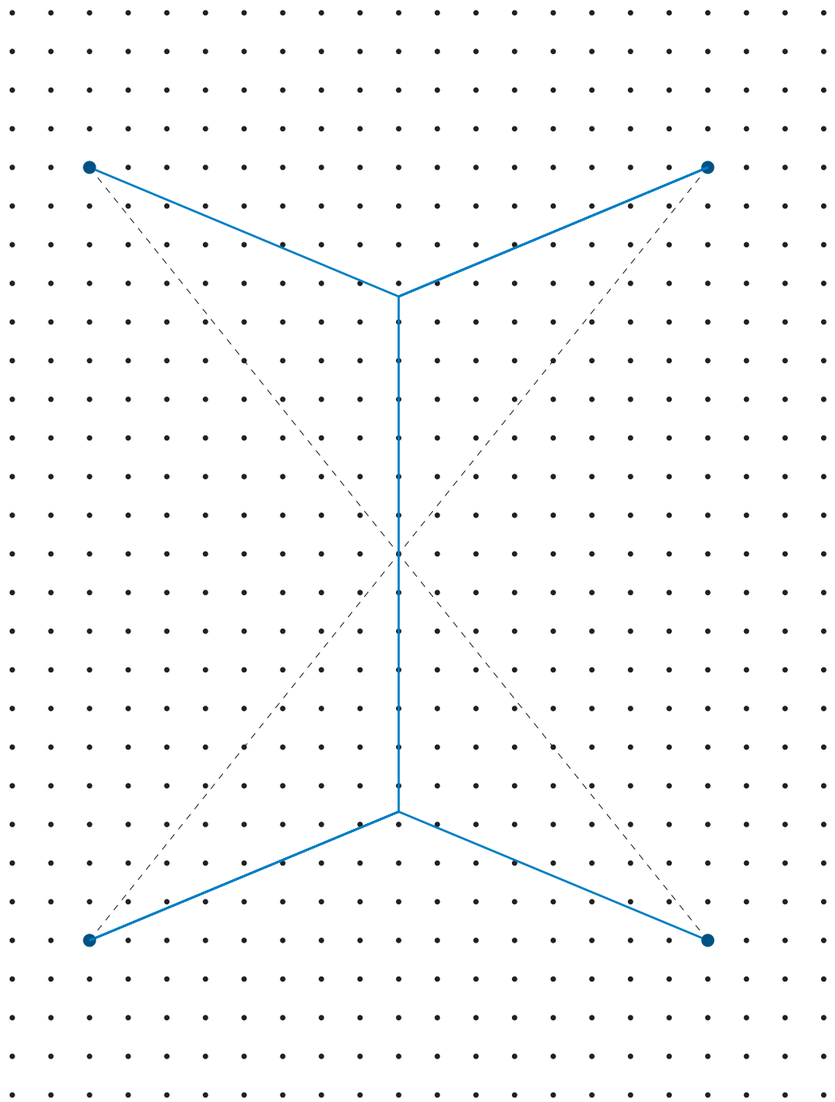


Best result: . . . . . .

Worksheet #17. Steiner Problem Solution.

Names: . . . . . .

Exercise 1. Draw the best route that you found and note your result below.



You can see solutions to two triangular Steiner problems here.

Education

SLIDES / HANDOUTS

6. SLIDES / HANDOUTS

An electronic version of the slides is available on the web page

http://livecomputing.org/slides/

5.1 Turing Test Conversations

The Turing Test Conversation #1

A: Hello I'm Ronan. What is your name?

B: My name is Rose.

A: How many children do you have?

B: It used to be two, but now it's four.

A: Do you prefer red or white wine?

B: I prefer red .

A: I like Waiting for Godot. What is your favorite play?

B: My favorite Shakespeare play is Romeo and Juliet . It shows the power of love. How about you?

Comment. This is a conversation between human (A) and a computer chat bot Rose (B) written by Bruce Wilcox. It is taken from a transcript of 2013 Loebner competition [transcripts].

The Turing Test Conversation #2

A: What is the name of Bart Simpson's baby sister?

B: I can't remember.

A: What do you think of Roald Dahl?

B: He writes funny books.

A: Are you a computer?

B: Are *you* a computer?

A: What is next number in the sequence 3,6,9,12,15?

B: 18.

A: What do you think of nuclear weapons?

B: Nuclear weapons are very dangerous and should be not used.

A: Do you like school?

B: Yes, I like school.

Comment. This is a conversation between two humans, but one human is pretending to be a computer.

The Turing Test Conversation #3

A: Men are all alike.

B: In what way?

A: They're always bugging us about something specific or other.

B: Can you think of a specific example?

A: Well, my boyfriend made me come here.

B: Your boyfriend made you come here?

A: He says I'm depressed much of the time.

B: I am sorry to hear you are depressed.

A: It's true. I'm unhappy.

B: Do you think coming here will help you not to be unhappy?

A: I need some help, that much seems certain.

B: What would it mean to you if you got some help?

A: Perhaps I could learn to get along with my mother.

B: Tell me more about your family.

Comment. This is . . .

The Turing Test Conversation #4

A: People get on my nerves sometimes.

B: I am not sure I understand you fully.

A: You should pay more attention.

B: Suppose you should pay more attention?

A: You're entitled to your opinion.

B: What makes you think I'm entitled to my opinion?

A: It bothers me just to be around people.

B: Please go on.

A: I know the mob controls the big rackets.

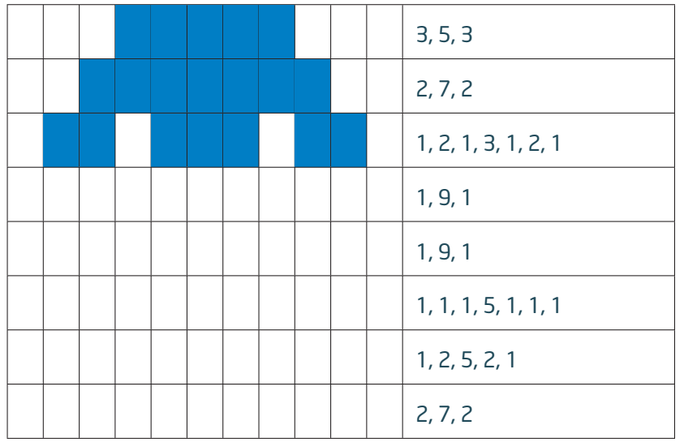
B: Who are you really talking about?

A: Have you ever really thought about the Mafia?

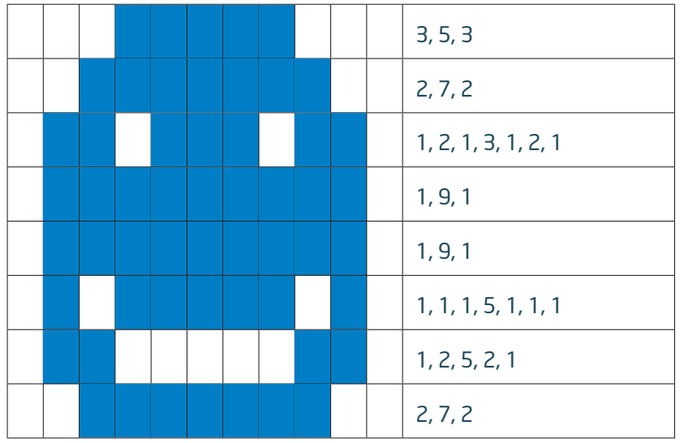
Comment. This is a conversation between two computers - PARRY (A) and DOCTOR (B) [?].

5.2 Image Representation

Exercise. Draw the missing part of the picture.



Solution.



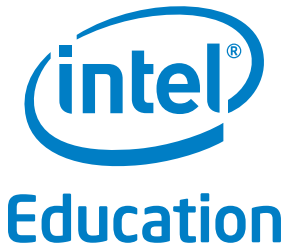
Education

EDUCATIONAL AIDS

7. EDUCATIONAL AIDS

A list of educational aids that accomany this Instructor's Handbook.

1. Binary numbers - cards with dots.
2. The adding machine.
3. Magnetic squares (40) for error correcting codes and 581 smaller squares for teams.
4. Alphabet wheels for ciphers.
5. Treasure boxes.
6. Weighting scale.
7. L-trominoes.
8. Cups and ping-pong balls for binary search.
9. Cards with numbers.
10. Sorting network.
11. Four decks of cards.
12. Towers of Hanoi.
13. TSP Boards (one large, five small).
14. Eliza Chatbot.


intel
Education