ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)
Vol. 2, Issue 2, Ver. 3 (April - June 2014)
International Journal of Advanced Research in Computer Science & Technology (IJARCST 2014)

Hands–on Gadgets to Facilitate Algorithmic Thinking for Beginners

S.R. Subramanya
School of Engineering and Computing, National University, San Diego, CA

Abstract
Algorithms are central to computing. With the use of computers in almost all areas of our lives, development of algorithmic solutions to problems is very crucial. The comprehension of the nature of problems and their algorithmic solutions is extremely important not only to students and professionals of Computer Science, but also in other domains. This paper presents several hands–on gadgets which have been developed to facilitate novices in programming and computing to improve the comprehension of a few representative algorithms, and as stepping stones to algorithmic thinking and programming. These gadgets also act as motivations for persons who may not have mathematical or computing background, but are interested in problem solving and developing programs.

Keywords
Algorithms; Hand–on gadgets; Algorithmic solutions; Algorithm comprehension; Algorithmic thinking

I. Introduction

Algorithms are at the heart of all computing. In the increasingly information and computation driven world, it is important to have a good grasp of algorithmic techniques for the development of solutions to problems in various domains such as Engineering, Sciences, Healthcare, Manufacturing, Management, Transportation, Urban Planning, Banking and Finance, Retail/Commerce, etc. For a beginner in Programming/Computing, it is extremely important to develop the skills of systematic development of precise steps of solutions to problems.

The steps in the algorithmic solution to a given problem consists of: (a) clearly stating the problem; (b) evaluating and choosing the appropriate algorithm design technique suited to the problem; (c) developing the solution using the algorithm; (d) proving the correctness of the solution; (e) analysing the solution complexity; (e) implementing the solution in a program; (f) running and testing the program. One of the ways of conveying the notions of algorithmic solutions for beginners and for non-specialists in Computer Science, as proposed in several studies, is use of appropriately designed hands–on gadgetry. Participants working individually and/or in groups get an insight into some problems and the complexities of their solutions.

II. Prior Work

In this section, we present several samples of work done in the area of the use of non–computer means to engage and teach students the skills of algorithmic thinking and problem solving.

The term computational thinking was introduced in [1], where computational thinking is described as consisting of a variety of elements drawn from Computer Science. Among other things, it involves solving problems, designing systems, reformulating a seemingly difficult problem into one we know how to solve, thinking recursively, using abstraction and decomposition, choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable, using heuristic reasoning to discover a solution, and several others. A variant, algorithmic thinking [2], are key abilities that can be learned independently from programming. It is shown in [2] that algorithmic thinking (a key ability in informatics) can be developed independently from learning programming, by the use of problems that are not easy to solve but have an easily understandable problem definition.

There have been several efforts in teaching computational/algorithmic thinking and programming without the use of computers, but by several paper–and–pencil methods and/or hands-on gadgets/tactile methods. A new technique for implementing educational programming languages using tangible interface technology, which makes use of inexpensive and durable parts with no embedded electronics or power supplies, is described in [3]. Programs are created in offline settings (on the desks or on the floor), and a portable scanning station is used to compile the code. It also describes an initial case study with children. Issues related to learning with tangible interfaces, suggesting that more empirically grounded research is needed to guide development is provided in [4].

The algorithmic thinking can be taught and carried out even without the use of computers, by just the use of pen–and–paper. A major aspect of the Australian Informatics Competition (AIC), which has a core focus on algorithms, is a pen–and–paper event, which is described in [5]. In addition to multiple choice questions (over algorithmic tasks, logic tasks, tracing tasks, and analysis tasks), the AIC has three–stage tasks requiring the use of algorithmic thinking by posing problems of increasing size.

Some of the perceptual and manipulative properties of physical objects with respect to their effect on the problem space and children's use of physical materials to solve numerical problems, and comparative performance using virtual materials is given in [6]. Tangible User Interfaces (TUIs) interlink the digital and physical worlds which have potential to enhance the way in which people interact with and leverage digital information. A history of tangible user interfaces, conceptual foundations of TUIs, a survey of application domains, review frameworks and taxonomies, and methods and technologies for designing, building, and evaluating TUIs is given in [7].

Critical thinking ability is gained through a variety of means. The thinking skills enable us to do computation effectively. Increased hands-on computation increases computational thinking. [8] provides a definition of computational thinking and connects the (potential) thinking elements to the known thinking paradigms.

Suitable environments with tangible objects and easy to understand problems has been shown to motivate the young to learn the first concepts of algorithms. A learning scenario targeted for primary school children using tangible objects facilitating a variety of interesting tasks to learn basic concepts of algorithmic thinking is presented in [9]. It also shows a smooth transition from tangible objects to a virtual Scratch/BYOB environment to help young learners to learn the first steps in understanding virtual

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

environments and programming concepts.
[10] identifies (a) situations in which tangible interaction offers advantages for learning, (b) situations in which tangible interaction is less useful, and (c) a hybrid approach—one that offers teachers and learners the flexibility to select the most appropriate interaction style to meet the needs of a specific situation.
A novel tangible, playful object called TimeBlocks, to facilitate communication about concepts of time with young children is presented in [11], which consists of a set of cubic blocks that function as a physical progress bar. Parents and children can physically manipulate the blocks to represent the concept of time. It also describes a range of observed insightful novel uses of TimeBlocks.
Multimodal interfaces (tablets, touch tables, and tangibles) have been suggested to have great potential to support children's learning and problem solving in spatial domains. However, there is little theoretical or empirical work identifying specific causes for many of the claimed benefits. Summaries of theoretical mechanisms explaining the benefits, and how the specific features of tangible interfaces might support or enhance these mechanisms are given in [12].
`Digital manipulatives' integrate digital technology into physical objects which have the potential to create new educational materials. [13] summarises the representational advantages of the manipulatives under (a) offloading cognition—where manipulatives may help children by freeing up valuable cognitive resources during problem solving, and (b) conceptual metaphors––where perceptual information or actions with objects have a structural correspondence with more symbolic concepts.
An approach that has been used in a Algorithms and Data Structures course, based on using a 'brick–box' (basic elements) to build 'buildings' (whole algorithms), without computers, by just writing on paper using Czech meta–language (based on Pascal) is described in [14].
In [15] (hands–on) assembling and programming the Lego robot, door alarm activity, and a couple of others, is described whose objectives are to introduce and kindle interest in students to computing.
An educational tangible user interface called AstroGrasp which facilitates collaborative hands–on learning of basic astronomy concepts of eclipses and the cause of seasons is presented in [16]. AstroGrasp allows users to explore the concepts of eclipse and seasons by physically manipulating hemispherical tokens of the Earth and the moon. The physical manipulation and continuous real–time digital feedback provided by AstroGrasp is expected to provide providing an effective, constructive and engaging learning experience.
A study exploring children's opinions and preferences for a tangible interface and a graphical interface to program a Lego NXT robot is given in [17]. The tangible interface comprised of 46 cube-shaped blocks representing simple programming structures which can be interconnected to form the programming code. The graphical interface consisted of on-screen icons similar in appearance and operation as with the tangible blocks. The quantitative and qualitative analyses leading to conclusions of the preferences by the children of different age groups, is given therein.

III. Hands–On Gadgets

There have been several efforts in teaching computational/ algorithmic thinking without the use of computers, but by several paper-and-pencil methods and/or hands-on gadgets/ tactile methods. In our study, we have developed a few simple gadgets (made out of wood, cardboard) which model problems to be solved. Having used these in several workshops on algorithmic thinking, the informal feedbacks have shown increased motivation to solve problems and getting a better feel for the nature of the problem and the solution.
In this section, we present descriptions of an assortment of a few problems, the descriptions of the corresponding hands–on gadgets which have been developed, the task performed by the participant(s) on the gadgets. For the sake of completion, the algorithmic solutions to the problems and their complexities are also given.

A. Knapsack Problem

Problem statement: Given a set of N objects with weights $w_1, w_2, \ldots \ldots w_N$, and values $v_1, v_2, \ldots \ldots v_N$, and a knapsack with capacity C, the problem is to determine the (subset) of objects that can be packed in the knapsack such that the sum of the values of the objects is the maximum possible subject to the condition that the sum of their weights is less than or equal to C.
Gadget description: The gadget consists of a rectangular base board with a wide groove at the centre, which represents the knapsack. There are rectangular pieces of different lengths with different numbers printed on them, which represent the objects. The lengths represent the weights of the objects, and the numbers on them represent the values of the objects. The length of the base board represents the capacity of the knapsack. The widths of the pieces are all the same, and are such that they fit snugly in the groove of the base board. These are shown in Fig. 1.

Fig. 1: Base board and rectangular pieces for the knapsack problem

Task: The task is to stack the pieces in the column such that the stack of pieces does not protrude out of the top boundary of the base board, while maximizing the sum of the values of the pieces.

Problem Solution: For extremely small instances of the problem (smaller problem sizes), exhaustive enumeration could be done and the optimal solution can be picked. However, this quickly becomes impractical even for small (say $N = 6$) problem sizes. A greedy solution yields suboptimal solutions, but can give results quickly even for large problem sizes. Using the greedy scheme, the object with the highest value per unit weight is first selected

if it is feasible, *i.e.*, adding it to the knapsack will still ensure the total weight of the objects in the knapsack is within its capacity, and the process is repeated.

Complexity: Determining the values per unit weight of the N objects takes O(N) time. Assuming K of the N objects are part of the feasible solutions, determining those 'feasible' objects with the highest value per unit weight, the next highest value per unit weight, and so on takes O(KN) time, which is the overall complexity.

B. Pancake Sorting

Problem statement: Given a set of pancakes of varying sizes which are stacked one on top of the other, (as shown in Fig. 2) , and given a spatula which can be inserted under any pancake and the whole stack of pancakes above the spatula can be flipped over, it is required to determine the number of flips required (in the worst case) in order to arrange the pancakes in sorted order with the largest pancake at the bottom.

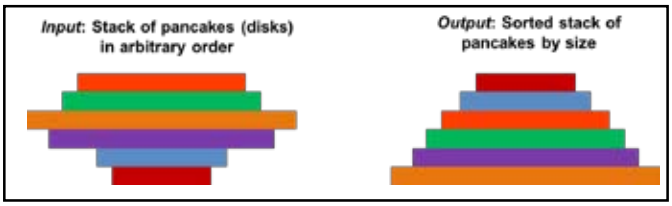

Fig. 2: Pancake sorting problem

Gadget description: The gadget consists of a base and a peg, and several coloured concentric disks of different colours and with increasing sizes, as shown in Fig. 3. The disks have holes at the centre and are stacked on top of the other.

Fig. 3: Concentric disks of varying sizes stacked on a peg for the pancake sorting problem

Task: Initially, the disks are stacked randomly, in no particular order. The task is to determine the sequence of moves where a chosen disk and all disks above it are removed, flipped over, and put back onto the peg, such that the disks are arranged in order of sizes with the largest one at the bottom.

Problem Solution: The spatula is placed below the largest pancake assuming that it is not yet in its final position. It is flipped over, after which the largest one is on the top. Then, the spatula is placed below the bottommost one, and flipped over, which results in the largest one going to the bottom. At this point, the largest one would have taken its final position. Next, the spatula is placed below the second largest pancake and flipped over, followed by placing the spatula above the largest one, and flipping over. This brings the second largest one above the largest one, which is its final position. This process is repeated.

Complexity: Note that the above process of flipping leads to the recurrence relation: $F(N) = F(N-1) + 2, N > 2$ and initial condition $F(2) = 1$. Solving this will give $2N - 3$ as the (worst case) number of flips.

C. Team Photo

Problem statement: There are two teams of N players and the players are of different heights. The teams need to be photographed with the teams lined in two rows and all members of team should be in the same row, and no player in the front row should be taller than the player right behind. What is a condition (constraint) that needs to be satisfied to ensure that a photograph could be taken?

Gadget description: Each player representation consists of cardboard (or wooden) cut–out. The cut–outs are of varying lengths corresponding to players of different heights. The cut–outs corresponding to players of a team are of the same colour (uniform).). These are shown in Fig. 4.

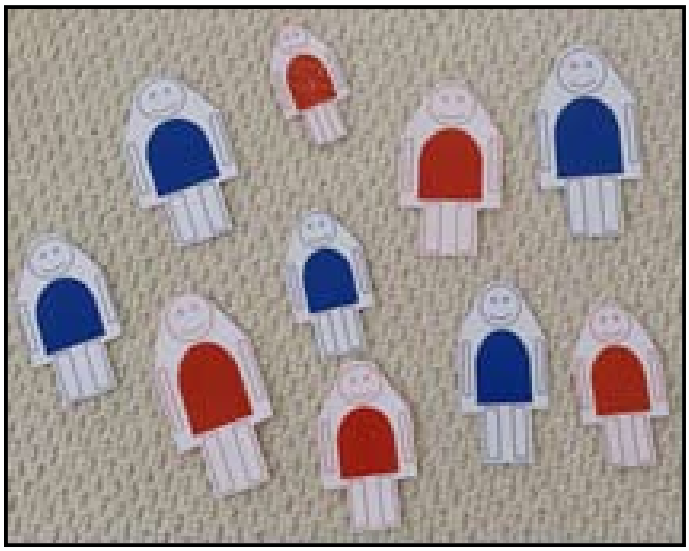

Fig. 4: Cardboard pieces for the team photo problem

Task: The task is to arrange the cut–outs such that all the ones of the same colour are in the same row, and no one in a front row is of larger length than the corresponding one in the back row.

Problem Solution: Suppose there are N members in each of the two teams. Represent the heights of members in two arrays A and B. Sort the arrays A and B in non–decreasing order. Then check if $A < B$ *(i.e.*, for all $1 \leq i \leq N$ if $A[i] < B[i]$) or $A > B$.

Complexity: Sorting the lists A and B are of complexity O(N log N). Checking to see if $A < B$ (or $A < B$) takes O(N) time. The overall complexity is thus O(N log N).

D. Post Office Location Problem

Problem statement: Given the distances between N villages which are located alongside a highway, the problem is to determine the

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

location of a post office (or some facility) in one of the villages such that the maximum distance from any village to the post office is the minimum possible.

Gadget description: It consists of rectangular ruler with some pictures of villages pasted on it at some places. There are markings on the ruler (to facilitate finding distances between the villages). These are shown in Fig. 5.

Fig. 5: (Part of a) wooden ruler for the post office location problem

Task: The task is to determine the desired location for the post office as described in the problem statement.

Problem Solution: First, the middle point between the first and the last villages is determined. If there is a village at the middle point, then it is the solution. Otherwise, the distance d_A between the rightmost village and the village to the left of the middle point (say A) and the distance d_B between the leftmost village and the village to the right of the middle point (say B) are determined. If $d_A < d_B$ then A is the solution, otherwise B is the solution.

Complexity: Computing the middle point takes constant time, and computing d_A and d_B requires constant time. Thus the overall complexity is O(1).

E. Telephone Bills and Payments Tracking

Problem statement: Given a list of telephone numbers to whom bills have been sent out, and another list of telephone numbers from whom payments have been received, the problem is to determine the telephone numbers for which payments have not been received.

Gadget description: There are two sets of wooden (cardboard) blocks. One set contains phone numbers in a colour (say Blue), and the other set contains phone numbers in a different colour (say Green), as shown in Fig. 6. The numbers in Blue represent the phone numbers for which bills have been sent. The numbers in Green represent the phone numbers for which payments have been received. The numbers in Green is a subset of the numbers in Blue.

Fig. 6: Wooden pieces with phone numbers for the telephone bills and payment tracking problem

Task: The task is to determine which numbers among the Blue coloured one are not present in the Green coloured ones. One way to go about it is placing all the Blue coloured numbers on the left side, and the Green coloured ones on the right side.

Problem Solution: Sort the two lists, B (Bills) and C (Payments). If ($B_i < C_j$), Add B_i to "Unpaid" list and increment i. Else if ($B_i == C_j$) Increment i and j. Keep doing until one or both lists are exhausted. If C is exhausted, put the remainder of B into "Unpaid" list.

Complexity: If the list B has N numbers and list C has M numbers ($N > M$), then sorting the lists B and C are of complexity O(N log N) and O(M log M), respectively. Checking for numbers common to both B and C (thereby finding the numbers in C but not in B) is O(N+M). Thus, the overall complexity is O(N log N) + O(M log M) + O(N+M) = O(N log N).

F. Most Living Celebrities

Problem statement: Given a list of celebrities (ex. Nobel prize winners), each element of the list containing the name, date of award (DoA), and date of death (DoD, if deceased), among other data, the problem is to determine the period of time with the most number of living celebrities.

Gadget description: It consists of wooden (or cardboard) pieces with pictures and other details of celebrities, , as shown in Fig. 7 Although the only data needed to solve the problem are the date of the award and the date of death (in case of deceased), the pictures and other data has been added to make it engaging and fun.

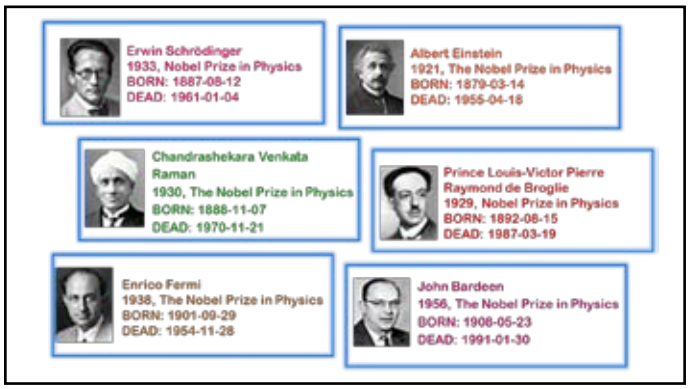

Fig. 7: A few samples of the blocks with details of celebrities (in this case, Nobel Prize winners in Physics) for the 'most living celebrities' problem

Task: The participants are required to use the blocks with the given information and determine the period of time with the most living celebrities.

Problem Solution: On a timeline, mark the dates of award with opening parentheses, and the dates of death with closing parentheses. Then, with an initial value of a variable **`count`** of zero, going from left to right on the marked timeline, increment the **`count`** whenever a "(" is encountered, and decrement the count whenever a ")" is encountered (as shown in Fig. 8). Then, scan the marked timeline to determine the interval whose beginning has the highest **`count`** value, which is the required solution.

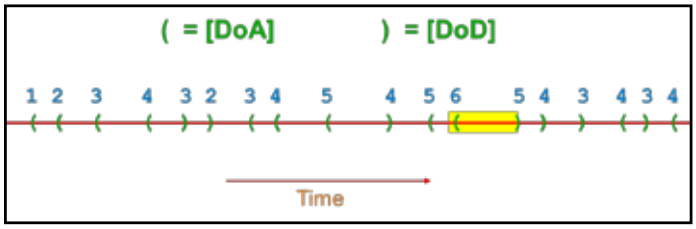

Fig. 8: Model for the solution of 'most living celebrities' problem

Complexity: Assuming that the list length is N. Sorting (DoA, DoD) takes O(N log N) time. Merging the DoA and DoD takes O(N log N) time. Subsequently, scanning the timeline to find the interval starting with the highest value takes O(N) time. Thus, the overall complexity is O(N log N).

G. Tiling for the Mosaic

Problem statement: Given a picture (colour or black & white), and given multiple copies of K distinct small thumbnail pictures – the *tiles*, to compose a mosaic of N x M tiles such that the mosaic is (visually) as close to the given picture as possible.

Gadget description: The hands–on gadget consists of a photograph and a collection of small thumbnail pictures, as shown in Fig. 9

Task: For each of the small thumbnail pictures, determine the position which 'best fits' in the mosaic such that, after the mosaic is completed, it best matches the given picture. The definition of 'best fits' is that the colours (or grey levels) of the thumbnail image is as close as possible to the corresponding position in the given picture per a given distance measure.

Problem Solution: For any tile at position (i, j) of the picture of N x M tiles, determine the closest match from among the thumbnail pictures which, when placed in that position, would make the mosaic visually as close to the picture as possible. The definition of 'closest match' is usually in terms of a suitably defined distance metric.

Complexity: Assuming the size of a tile (and the thumbnail picture) is a x b, the distance computation requires O(a x b) time. Assuming the thumbnail pictures being in no particular order, selecting the best matching thumbnail picture for a given tile position in the mosaic will have to look at all the K (Brute Force) which takes O(K) time. Thus for the whole mosaic, it takes O(NMK) time. Organizing the K thumbnail pictures as leaves of a (binary) tree, and using the 'average' of two child nodes for the parent node, searching for the appropriate thumbnail pictures for the whole mosaic would take O(NM log K) time.

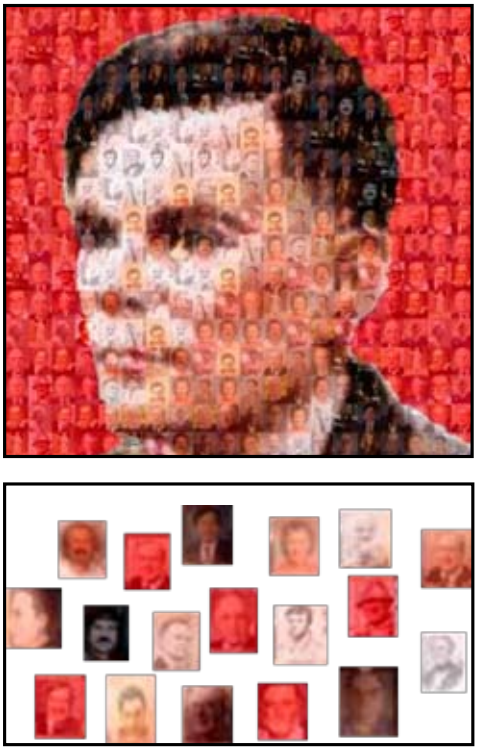

Fig. 9: Mosaic (above) and a few sample tiles (below) for the tiling problem

H. Coin Row Problem

Problem statement: There is a row of N coins with values $c_1, c_2,..., c_N$, not necessarily distinct. The goal is to pick up the maximum amount of money subject to the constraint that no two adjacent coins in the row can be picked up.

Gadget description: It consists of circular plastic pieces with denominations marked, and a rectangular bar with depressions on which the plastic pieces are placed, as shown in Fig. 10.

Fig. 10: Pieces with 'denominations' for the coin row problem

Task: The task is to pick the pieces which add up to the maximum value marked with the condition that no two adjacent ones are picked.

Problem Solution: The solution can be expressed as a recurrence relation. Let $F(N)$ be the maximum amount that can be picked up from the row of N coins. Partition the permissible coin selections into two groups: (a) those without last coin and (b) those with last coin. Then, we can develop the recurrence $F(N) = \text{MAX}\ \{c_N + F(N-2),\ F(N-1)\}$ for $N > 1$, with the initial conditions $F(0) = 0$, $F(1) = c_1$. The actual coins to be picked up is determined by back– tracing.

Complexity: The complexity of back– tracing over the row by applying the MAX function given above takes O(N) time.

I. Robot Coin Collection Problem

Problem statement: Several coins are placed on a board of $N \times M$ cells. A robot, located in the upper left cell of the board, needs to collect as many of the coins as possible and bring them to the bottom right cell. On each step, the robot can move either one cell to the right or one cell down from its current location.

Gadget description: It consists of a rectangular cardboard piece with cells marked. Several circular cardboard pieces are placed (randomly) on several cells (with some cells left empty), as shown in Fig. 11.

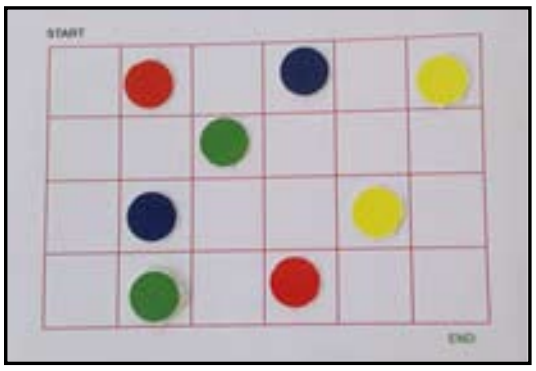

Fig. 11: Rectangular board with circular pieces ('coins') for the robot coin collection problem

Task: Starting at the left–left cell move successively one cell down or one cell to the right and reach the bottom–right cell. During this move, while on a cell with a circular piece, it is picked up. The task is to trace and find the path which leads to the maximum number of pieces collected.

Problem Solution: This problem is amenable for a neat recurrence relation. $F(i,j)$: largest number of coins the robot can collect and bring to cell (i,j) in the ith row and jth column. $F(i, j) = MAX\ \{F(i-1, j),\ F(i, j-1)\} + c_{ij}$ for $1 \le i \le N$, $1 \le j \le M$, where $c_{ij} = 1$ if there is a coin in cell (i,j), and 0, otherwise. The initial

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

conditions are, $F(0, j) = 0$ for $1 \leq j \leq M$ and $F(i, 0) = 0$ for $1 \leq i \leq N$. The required solution is $F(N, M)$. The actual path leading to the solution is obtained using back– tracing.

Complexity: The complexity of obtaining the maximum number of coins collected and the path taken is $O(N + M)$.

IV. Conclusions

Algorithms are at the heart of all computing. In the increasingly information and computation driven world, a good grasp of algorithmic techniques is extremely important for the development of elegant and efficient solutions to problems in various domains. Also, algorithmic thinking is considered one of the 21st century skills and is important across several workplaces. This paper described the use of several hands–on gadgets and the tasks required of participants to work on. It also provided the solutions to the problems and their complexities. Based on qualitative and informal feedback, the use of hand–on gadgets has been shown to generate interest in developing algorithmic solutions to problems, and to facilitate a good understanding of the problems and the development of required solutions.

V. AcknowledgEment

The author wishes to thank Mr. Timothy Wade, Sr. Security Integration Engineer at Leidos, for help in getting several of the gadgets made.

References

[1] *J. Wing, Computational thinking. Communication of the ACM, 49(3), 2006, 33–35.*

[2] *G. Futschek, Algorithmic thinking: the key for understanding computer science, Proceedings of the 2nd International Conference on Informatics in Secondary Schools: Evolution and Perspectives (ISSEP2006), 159–168.*

[3] *M.S. Horn and R.J.K. Jacob, Designing tangible programming languages for classroom use, Proceedings of the 1st international conference on Tangible and Embedded Interaction (TEI'07), 159–162.*

[4] *P. Marshall, Do tangible interfaces enhance learning?, Proceedings of the 1st international conference on Tangible and Embedded Interaction (TEI'07), 163–170.*

[5] *B.A. Burton, Encouraging Algorithmic Thinking Without a Computer, Olympiads in Informatics, 2010, Vol. 4, 3–14.*

[6] *A. Manches, C. O'Malley, and S. Benford, Physical manipulation: evaluating the potential for tangible designs, Proceeding TEI '09 Proceedings of the 3rd International Conference on Tangible and Embedded Interaction, 77–84.*

[7] *O. Shaer, E. Hornecker, Tangible User Interfaces: Past, Present, and Future Directions, Journal of Foundations and Trends in Human–Computer Interaction archive, Volume 3 Issue 1–2, January 2010, 1–137.*

[8] *C. Hu, Computational Thinking – What It Might Mean and What We Might Do About It, Proceedings of ITiCSE'11, June 27–29, 2011, Darmstadt, Germany, 223–227.*

[9] *G. Futschek and J. Moschitz, Learning algorithmic thinking with tangible objects eases transition to computer programming, Proceeding ISSEP'11 Proceedings of the 5th international conference on Informatics in Schools: Situation, Evolution and Perspectives, Springer, October 26–29, 2011, Bratislava, Slovakia, 155–164.*

[10] *M.S. Horn, R.J. Crouser, and M.U. Bers, Tangible interaction and learning: the case for a hybrid approach, Journal of Personal and Ubiquitous Computing, Volume 16 Issue 4, April 2012, 379–389.*

[11] *E. Hayashi, et. al., TimeBlocks: mom, can I have another block of time, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2012, 1713–1716.*

[12] *A.N. Antle, Knowledge gaps in hands–on tangible interaction research, Proceedings of the 14th ACM international conference on Multimodal interaction, 2012, 233–240.*

[13] *A. Manches and C. O'Malley, Tangibles for learning: a representational analysis of physical manipulation, Journal of Personal and Ubiquitous Computing, Volume 16 Issue 4, April 2012, 405–419.*

[14] *E. Mlkova and A. Hulkova, Algorithmic and Logical Thinking Development: base of programming skills, WSEAS Transactions on Computers (E–ISSN: 2224–2872), Issue 2, Volume 12, February 2013, 41–51.*

[15] *N. Swain, W. Moses, J.A. Anderson, and C.T. Davis, Computational Thinking in K–12 Schools Using Hands–on Activities, 120th ASEE Conference and Exposition, paper id #6400, Atlanta, GA, June 2013.*

[16] *H. Agrawal and K. Sorathia, AstroGrasp: a tangible user interface for teaching basic astronomy concepts, Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction, 2013, 123–128.*

[17] *T. Sapounidis and S. Demetriadis, Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences, Journal of Personal and Ubiquitous Computing, Volume 17 Issue 8, December 2013, 1775-1786.*