IJournals: International Journal of Software & Hardware Research in Engineering
ISSN-2347-4890
Volume 4 Issue 6 June, 2016

Hands–on Gadgets to Facilitate and Augment Algorithmic Thinking and Problem Solving for Beginners

S.R. Subramanya
School of Engineering and Computing
National University
San Diego, CA 92123
ssubramanya@nu.edu

ABSTRACT

Algorithms are at the heart of all computations performed in the software used in the various domains such as, Engineering, Sciences, Healthcare, Transportation, Urban Planning, Banking and Finance, Retail/Commerce, etc. In the increasingly information driven world, it is essential for students and professionals not only of Computer Science, but also of other domains, such as Engineering, Biological Sciences, Physical Sciences, Economics, Finance, Health Sciences, etc., to have the capabilities of critical thinking, problem formulation and analysis, and algorithmic development of solutions to numerous real–world and societal problems. This paper presents samples of some hands–on gadgets which we have developed to facilitate novices in computing to improve the comprehension of a few representative classical problems, and as stepping stones to algorithmic thinking. Informal qualitative feedback has shown the effectiveness of the hands–on gadgets for the beginners, in facilitating a good understanding of the problems, and in the development of the required solutions.

Keywords: Algorithms, Algorithmic thinking, Hands-on gadgets

1. INTRODUCTION

Algorithms are at the core of all computing. In this increasingly competitive, information–driven and globalized economy, it is extremely important even for students in disciplines other than Computer Science to be conversant with computing and information technology concepts and applications. In fact, these are frequently referred to as part of 21st century skills. Mere fluency with technology and use of computers are not adequate skills in the future, but they need to be augmented with skills for effectively solving information–rich, information–driven, real–world and societal problems. It is advantageous to have a good grasp of algorithmic techniques which they can later apply effectively to solve problems in their chosen domains of specializations such as Engineering, Sciences, Healthcare, Transportation, Urban Planning, Banking and Finance, Retail/Commerce, etc. Growths in Information Technology driven economies together with globalization has been resulting in shifts in occupations. Increasing economic importance of digital commerce and digital content, and IT in the workplace are seen as drivers of changing employment landscape [1]. These require familiarity with technologies combined with analytical and problem-solving skills. In [2], a model with five increasingly mature levels developed by SAS called the *Information Evolution Model* to help organizations think about how they use information is presented. Numerous sources (as an example, [3]) have compiled lists of "21st century skills" which the students need in order to succeed in work. Among other things such as global awareness, financial, economic, business and entrepreneurial literacy, skills such as critical thinking and problem solving skills, communications skills, creativity and innovation skills, collaboration skills, information and communications technology literacy are considered important. The study [4], confirms the positive connection between the development of the competencies known as 21st century skills – critical thinking, real world problem solving, collaboration and communication skills – and

future job success.

STEM (Science, Technology, Engineering, Math) competencies are deemed very important in today's increasingly technology–driven world. According to a recent report from the Government Accountability Office, the U.S. government spends more than US $3 billion each year on 209 STEM-related initiatives overseen by 13 federal agencies. In addition, major corporations are collectively spending millions to support STEM educational programs. Also, every U.S. state has its own STEM initiatives. Despite these, there have been two alarming trends – not many students enrolling in STEM related studies, and that the universities are not delivering STEM graduates with the qualities demanded by employers. The reports [5, 6] analyze and note that the biggest challenges facing STEM provision seem to be in how individuals progress into the labor market. There is a sense of urgency to draw more students into STEM disciplines, and to strengthen the quality of STEM degrees and to boost industry and entrepreneurialism skills for graduates.

In this paper, we present the use of hands–on gadgets to evoke interest among the beginners, without prior programming experience, in developing algorithmic solutions to problems. The steps in the algorithmic solution to a given problem consists of the following: (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 the use of appropriately designed hands–on gadgetry. We describe several of the gadgets which have been specifically designed for some (classical) algorithmic problems. Based on informal, qualitative feedback, the use of hand–on gadgets has demonstrated evocation of interest among the participants. Working hands–on with the gadgets facilitates (a) concretization of abstract problem statements; (b) better understanding of the problem; (c) exploration of the solution space; (d) insights into algorithmic thinking and solution development.

2. BACKGROUND AND RELATED WORK

In this section, we present several work related to the use of hands-on gadgets which facilitate computational/algorithmic thinking. The term *computational thinking* was introduced in [7], 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*, comprises key abilities that can be learned independently from programming. It has been shown 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 [8].

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), is described in [9]. 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 [10].

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. 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 [11].

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 [12].

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 [13]. 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 environments and programming concepts. Hands–on activities of assembling and programming the Lego robot, door alarm activity, and a couple of others, whose objectives are to introduce and kindle interest in students in computing is described in [14].

3. SAMPLES OF HANDS–ON GADGETS

In our study, we have developed a few simple gadgets (or artefacts) made out of wood, cardboard, etc. where each of the gadgets models a problem (or class of problems). The problem to be solved, the operation of the gadget (as applicable), the constraints, and the desired solution are clearly stated. Using the gadget, and keeping in mind the constraints, the solution needs to be obtained. This process gives the users/participants a good feel about the nature of the problem, the constraints, and more importantly, the need for a systematic exploration of the solution space. Use of these gadgets in several workshops on algorithmic thinking, along with the informal feedbacks have shown that the gadgets provide increased motivation to solve problems and provide a better feel for the nature of the problem and the solution.

In this section, we present descriptions of several problems, the descriptions of the corresponding hands–on gadgets which they have been developed, and the task performed by the participant(s) on the gadgets. For the sake of completion, the algorithmic solutions to the problems are also given.

3.1 Knapsack problem

Problem statement: Given a set of N objects with weights w_1, w_2, … … w_N, and values v_1, v_2, … … 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 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 (not already in the knapsack) 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. This process is then repeated. The Pseudocode of the process is given below.

```
begin
  i = 1; avail_capacity = M;
      Sort the objects in non-
      decreasing order based on
      value/weight ratio and label
      them 1 to N.
  while (i ≤ N) {
    Pick object_i.
    if (weight_i ≤ avail_capacity){
       Add object_i to Knapsack.
       avail_capacity ←
      avail_capacity - weight_i;
    }
  }
end
```

3.2 Collating disks of like colors

Problem statement: Given N disks consisting of some red and blue ones, in some arbitrary order in a row, it is required to arrange them such that all blue disks are followed by all red disks, using the minimum number of operations. The permitted operation is the swap operation – concurrent exchange of two disks.

Gadget description: The gadget simply consists of several red and blue wooden disks, placed in some mixed order along a line as shown in Fig. 2.

Fig 2: Red and blue disks for the problem of collating disks of like colors

Task: Initially, the disks are placed along a line, in no particular order. The task is to arrange them such that all the blue disks are together, followed by all the red disks. The constraint is that it should be accomplished using the minimum number of swap (concurrent exchange of two disks) operations.

Problem Solution: We could use a (virtual) "marker" to indicate the current disk under consideration. Thus we use two markers – ***L*** (left) and ***R*** (right), which are initially at the leftmost and at the rightmost disks, respectively. If the disk pointed to by *L* is blue, we move the marker one place to the right, and keep repeating until the disk pointed to by it is not blue. Similarly, as long as the disk pointed to by the marker *R* is red, keep moving the pointer successively by one position left. When the disk pointed to by *L* is red and that pointed to by *R* is blue, swap the disks. Continue this process as long as *L* and *R* do not overlap.

This process has been captured in the Pseudocode below. It is easy to see that the variables ***i*** and ***j*** correspond to the *L* and *R* markers.

```
begin
  i = 1; j = N;
  while (i < j) {
    while (Disk[i] == "Blue"
            AND i < j) i++;
    while (Disk[j] == "Red"
            AND i < j) j--;
       if (i < j)
         SWAP (Disk[i], Disk[j]);
  }
end
```

3.3 Dutch national flag problem

Problem statement: Given N disks consisting of some Red, White, and Blue, in some arbitrary order in a row, arrange them such that all Red disks are followed by all White disks, followed by all Blue disks using the minimum number of swaps. The allowed operations are color examination and swap of disks.

Gadget description: The hands–on gadget consists of several Red, White, and Blue wooden disks, placed in some mixed order along a line as shown in Fig. 3.

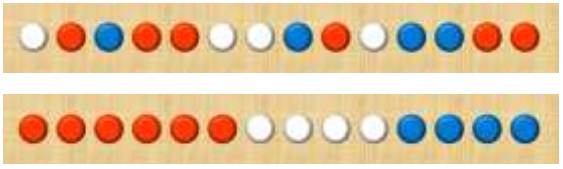

Fig 3: Red and Blue disks for the Dutch national flag problem (top: some initial state; above: final state)

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:

Initially the "All Red", "All White", and "All Blue" regions are empty (All marbles are in the "Mixed R, W, B section"). The Pseudocode of the process is given below.

1. If the leftmost disk in the "Mixed R, W, B" section is Red, swap it with the first disk after the "All Red" section, and move to the next disk

2. If the leftmost disk in the "Mixed R, W, B" section is White, do nothing, and move to the next disk

3. If the leftmost disk in the "Mixed R, W, B" section is Blue, swap it with the disk before the "All Blue" section, and move to the next disk

At each step (iteration), the "Mixed R, W, B" section is reduced by 1. The process is continued until the "Mixed R, W, B" section disappears.

3.4 Partition problem

Problem statement: The partition problem is the task of deciding whether a given multi-set S of positive integers can be partitioned into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2.

Gadget description: The gadget for this problem consists of several wooden strips of varying lengths, as shown in Fig. 4. There could be a few strips of equal lengths as well.

Fig 4: Wooden strips of various lengths for the partition problem

Task: The task is to divide the given set of strips into two subsets such that the sums of the lengths of the strips in each of the subsets are equal. The participants place the strips end-to-end along two lines and check whether they are of equal lengths. In a well-designed gadgetry, they have to try numerous combinations.

Problem Solution: The solution can be obtained by an exhaustive search approach. The subsets of strips are selected methodically, and checked to see if their combined lengths equals L. The Pseudocode of the process is given below.

```
begin

   Form the subsets S1 and S2 using
   the elements from the original
   set of numbers, S.

   (Note: S1 would be all possible
   subsets of N/2 elements, and S2
   would automatically be the
   subset of the remaining elements
   S-S1)

   Add the numbers in S1 and S2 and
   check if they are equal.

   Repeat the above steps until two
   subsets whose sums of elements
   are equal, or all possible
   subsets are exhausted.

end
```

This gadget can also be used for another (similar) problem, namely, given a multi-set S, to partition it the into two subsets S_1, S_2 such that the difference between the sum of elements in S_1 and the sum of elements in S_2 is minimized.

3.5 2D Bin packing

Problem statement: In the two-dimensional bin packing problem, we are given an unlimited number of finite identical rectangular bins, each having width W and height H, and a set of N rectangular items with width $\boldsymbol{w_j} \leq W$ and height $\boldsymbol{h_j} \leq H$, for $1 \leq j \leq N$. The problem is to pack, without overlap, all the items into the minimum number of bins.

However, we modify the original problem statement and state that, given a rectangular bin of width W and height H, and a set of N rectangular items with width $\boldsymbol{w_j} \leq W$ and height $\boldsymbol{h_j} \leq H$, for $1 \leq j \leq N$. The problem is to place all the items into the bin, without overlap, such that the uncovered area is minimized.

Gadget description: The gadget consists of a (wooden) base board with a rectangular groove at

the centre, which represents the bin. There are rectangular (wooden) pieces of different lengths and widths, which represent the objects. These are shown in Fig. 5.

Fig 5: Wooden base board and rectangular pieces of various sizes for the bin packing problem

Task: The task is to place the rectangular pieces in the groove of the base board such that they are non-overlapping, and the uncovered area is the minimum possible.

Problem Solution: This is a hard problem. There are several approximations. One of the simple ones is a first–fit scheme where the successively smaller pieces are tried in order. The other one is a best–fit scheme which attempts to minimize the 'fragmentation' and the uncovered area.

3.6 Sorting

Problem statement: To sort numbers by comparing a pair of numbers place rods in sorted (increasing) order of their lengths.

Gadget description: The hands–on gadget consists of a baseboard with rectangular slots where a rectangular rod could fit in, and set of rods of different lengths, as shown in Fig. 6. Initially, the rods are placed in some order (which is not necessarily in any order).

Fig 6: Wooden base board and rectangular pieces of various sizes for the sorting problem

Task: The rods need to be arranged in order of increasing lengths from left to right, using permitted moves, where each move consists of taking out two rods and swapping (exchanging) them.

Problem Solution: There are many different ways of going about this. One of the (simple) ways is to choose the smallest one which is not already in its final position *p*, and swap it with the one which is in position *p*. Repeat these steps until all the rods are in order. The Pseudocode of the process is given below.

```
begin
  L ← 1; R ← N;
  while (L < R) {
       k ← The index of smallest
       element of A[L..R].
       SWAP (A[L], A[k]);
       L ← L + 1;
  }
end
```

3.7 Matching nuts and bolts

Problem statement: Assortment of *N* nuts and *N* bolts of varying sizes, using the minimum number of trials, match the *N* nuts to *N* bolts (of varying sizes)

Gadget description: This is nothing but a pile of nuts and bolts of assorted sizes as shown in Fig. 7.

Fig 7: Assortment of nuts and bolts of various sizes for the 'matching nuts and bolts' problem

Task: To match each bolt with a matching nut (if it exists) using the *minimum* number of trials. A trial consists of picking a nut and a bolt and seeing if it matches. A systematic way of doing this consists of first ordering the nuts and bolts in non-decreasing order.

Problem Solution: Sort the *N* nuts and *N* bolts. Try the nuts and bolts for a match in order. The Pseudocode of the process is given below.

```
begin
  i ← 1; j ← 1;
       Sort the Nuts and Bolts in
       non- decreasing order and
       label them 1 to N
  while (i ≤ N AND j ≤ N) {
       if (Nut_i MATCHES Bolt_j) {
             Print ("Nut_i matches
             with  Bolt_j");
        i++; j++
       }
       else if (Nut_i SMALLER THAN
       Bolt_j)
          i++;
       else
          j++;
  }
end
```

3.8 Pipeline location

Problem statement: There are several oil wells in an area, and a straight oil pipeline is required to be laid out (running East–West) to transport the oil. Feeder lines (Spur pipelines) need to be laid out from the oil wells to the oil pipeline (running North–South). The problem is to determine the location of the pipeline such that the sum of the lengths of the spur pipelines is minimized.

Gadget description: This consists of shapes made of wood (or cardboard) as shown in Fig. 8. The red circular disks denote the locations of oil wells, the green strips represent the spur pipelines, and the Blue wide strip represents the main oil pipeline.

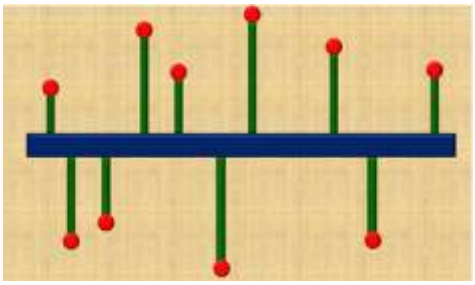

Fig 8: Wooden circular disks and rectangular strips of various lengths for pipeline location problem

Task: The locations of the oil wells (round red disks) are fixed. The spur pipelines (green strips) connected to the oil wells are placed perpendicular to the *X*–axis. The task is to place the main pipeline (blue strip) such that the sum of the lengths of the spur pipelines are minimized. A ruler is provided to measure the required lengths.

Problem Solution: Determine the median of the *Y*–coordinates of the locations of the oilwells. Placing the oil pipeline at the median would minimize the total lengths of the spur lines.

3.9 Shopping center locations

This ***Problem statement***: There are *N* villages on either side of a long segment of a highway. As a convenience to the people living there, a company wants to build shopping centers. What is the ***minimum number*** of shopping centers (and their locations) such that the distance from any village to a shopping center is no more than ***K*** miles?

Gadget description: The hands–on gadget consists of a long strip (of wood or cardboard) representing the highway, a set of small circular disks (of wood or cardboard) representing the villages placed on either side of the long strip, and a ruler, as shown in Fig. 9.

Fig 9: Wooden circular disks and a long strip for the shopping centers location problem

Task: Given a certain distance *K*, it is required to determine the smallest number of locations which would be the sites of shopping centers, such that the distance from any of the villages to a shopping center location is no more than *K*. A ruler is provided to measure the required lengths.

Problem Solution: Find the leftmost village which is not within *K* miles from a shopping center. Mark the location *K* miles to the right of such a village for the next shopping center. Repeat until no such village (*i.e.*, not within *K* miles from a shopping center) is remaining. The Pseudocode of the process is given below.

```
begin
   i ← 1; j ← 0;
   SC[j] ← V[i] + K;
   for i = 2 to N-1 do
      if (V[i] – SC[j] > K){
         j ← j + 1;
         SC[j] ← V[i] + K;
      }
   endfor
  Print ("Number of Shopping
  Centers:" j+1);
  Print ("The locations of the
  shopping centers:", SC[0..j]);
End
```

4. 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 presented the use of several hands–on gadgets which model specific (classical) problems. It described the tasks that the participants were required to work on in order to solve the problems using the gadgets. The participants were presented with the Pseudocodes of the solutions to the original problems. This methodology enables the participants to make a connection between the hands-on processes used with the gadgets and the solutions expressed as Pseudocodes. Based on qualitative and informal feedback, the use of hand–on gadgets has been shown to generate interest both among the beginners with no prior programming experience, as well as those with some programming experience, in developing algorithmic solutions to problems, and to facilitate a good understanding of the problems and the development of the required solutions.

5. REFERENCES

[1]. C. Anderson and J.F. Gantz, "Requirements for Tomorrow's Best Jobs – Helping Educators Provide Students with Skills and Tools they Need", IDC White Paper, Oct. 2013.

[2]. J. Goodnight, "Educating for Global Competitiveness – A white paper on education in the 21st century", SAS Institute, 2006.

[3]. B. Trilling and C. Fadel, "21st Century Skills – Learning for Life in our Times". John Wiley, 2009.

[4]. Gallup "21st Century Skills and the Workplace: A 2013 Microsoft Partners in Learning and Pearson Foundation Study", 2013. http://www.gallup.com/strategicconsulting/162821/21st-century-skills-workplace.aspx

[5]. C. Levy and L. Hopkins, "Shaping Up For Innovation: Are we delivering the right skills for the 2020 knowledge economy?", *The Work Foundation* (www.theworkfoundation.com), 2010.

[6]. H.B. Gonzales and J.J. Kuenzi, "Science, Technology, Engineering, and Mathematics (STEM) Education: A Primer", *Congressional Research Service*, August 2012.

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

[8]. 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.

[9]. 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.

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

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

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

[13]. 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.

[14]. 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.

.

