Activities designed and developed by
Laura McGinnis
Pittsburgh Supercomputing Center

Materials designed and prepared by
Anabelle Lee
Carnegie Mellon University

Additional materials based on activities from
Computer Science Unplugged
Created by Tim Bell, Ian H. Witten and Mike Fellows,
Illustrated by Matt Powell
http://csunplugged.org/
Using STAKES to support STEM

Numerous studies have shown the value of experiential learning in mastery of complex topics. The Pittsburgh Supercomputing Center (PSC) extends this principle to education in STEM (Science, Technology, Engineering, and Mathematics) by developing and deploying STAKES - Science and Technology Active Knowledge Experiences. STAKES are opportunities for students to explore and experience the impact of science and technology in active, rather than passive, ways.

Activities:

001
OPS - How Fast Can You Calculate?

002
The Megagrain - How Big is a Million?

003
Data Sets - Why Do We Care?

004
Color by Numbers - Image Representation

005
Twenty Guesses - Information Theory

006
Lightest and Heaviest - Sorting Algorithms

007
Beat the Clock - Sorting Networks

008
The Muddy City - Minimal Spanning Trees

009
The Orange Game - Routing and Deadlock

010
Marching Orders - Programming Languages

011
STEM in Life - Current Events

012
Careers in Supercomputing
How fast can you calculate?

--- OPERATIONS PER SECOND

Can you compute as fast as one operation per second? Most supercomputers calculate in trillions of OPS. However, since they calculate real numbers instead of single-digit integers, their unit of work is a Teraflop: a trillion floating point operations per second.

--- PARALLELISM

Multiple systems working together, i.e. working in parallel, can process a large amount of work in a much faster time, although this decrease in time isn’t necessarily proportional to the number of machines at work.

--- I/O BOTTLENECKS

Reading a problem (input) and writing down the answer (output) can slow down the time it takes to complete the task. This is called a "bottleneck".

--- ERROR SIGNALS

Not all systems have 100% accuracy. Error detection and correction also slow down the time it takes to solve a problem. The OPS also differs according to whether an error check is run for each operation, or run over the entire batch after the initial work is done.
THE MEGACRANIA: how big is a million?

--- PARALLELISM

A task can be sped up greatly by dividing it into parts and distributing the work across several different systems, i.e. working systems in parallel. However, the decrease in time isn't necessarily proportional to the number of systems that are working.

--- SCALE

"Scaling" refers to changes in size and/or complexity. The effect of a change in order of magnitude may have more impact than you expect. 1,000 grains of rice doesn't weigh much, and doesn't take up much space. One million grains of rice becomes a bit more significant: about 32 pounds. One trillion grains, a million million, would fill 187 tractor trailer trucks.

--- POWERS OF TWO

If you start with a pile of rice with one grain, then two, then four, and continuously double this number, it will increase very slowly at first, but start building up very, very quickly.
DATA SETS: why do we care?

How much is a trillion?

Computational tools allow scientists to model processes too large, too costly, or too complicated to investigate in other ways. Computational tools such as modeling and simulation can be considered a new way of doing science, along with experimentation to test a hypothesis.

As models and simulations become larger and more complex, larger and more complex computers are required to process the data. The largest computers are still too small to handle many of the large systems scientists want to study.

Try to match up the system or data set to its size. It's trickier than it looks!
match the quantity...  

- molecules in one mole (18g) of water  
  CREDIT: http://www.biology.ilstate.edu

- cells in the human body  
  CREDIT: Pittsburgh Supercomputing Center

- population of the Earth  
  CREDIT: http://public.boulder.ibm.com/neocerma/

- stars in the Milky Way  
  CREDIT: Astrophysics Data Facility at the NASA Goddard Space Flight Center

- atoms in the Tobacco Mosaic Virus  
  CREDIT: University of Illinois at Urbana-Champaign's Theoretical and Computational Biophysics Group

- surface area of the Earth in square miles  
  CREDIT: Lunar and Planetary Institute

...with the item

- 1,000,000 (one million or 1 x 10^6)

- 196,939,900 (197 million, or 197 x 10^6)

- 6,653,077,496 (6.653 billion, or 6.653 x 10^9)

- 1,000,000,000,000 (one trillion, or 1 x 10^{12})

- 10,000,000,000,000 to 100,000,000,000,000 (10 to 100 trillion or 10 to 100 x 10^{12})

- 602,000,000,000,000,000,000,000,000,000 (602 billion trillion or 6.02 x 10^{23})
Computer screens are divided up into a grid of small dots called pixels (picture elements).

In a black and white picture, each pixel is either black or white.

The letter “a” above has been magnified to show the pixels. When a computer stores a picture, all that it needs to store is which dots are black and which are white.

The picture above shows how a picture can be represented by numbers. The first line consists of one white pixel, then three black, the one white. Thus, the first line is represented as 1,3,1.

The first number always relates to the number of white pixels. If the first pixel is black, the line will begin with a zero.
Try recreating the original images off of these grids and coordinates! The first is easiest, and the last is the most complex.
Create Your Own Picture

Try creating your own coded image for a friend! Draw your picture on the top grid and when you’ve finished, write the code numbers beside the bottom grid. Cut along the dotted line and give the bottom grid to a friend to color in.
How much information is there in a 1000-page book? Is there more information in 1000 pages of a telephone directory, or in 1000 sheets of blank papers? If we can measure this, we can estimate how much space is needed to store the information. For example, you can still read many sentences that are missing their vowels, because there is not much “information” in vowels.

Because saving information using as little space as possible is important for computers, computer scientists usually don’t measure the value of information by letters or pages, but by how “surprising” the information is.

A famous American mathematician called Claude Shannon did many experiments using a game similar to “20 Questions” to research the value of information. He found that the amount of “information” contained in a message depends on what you already know. He called the information content of a message “entropy,” which depends not only on the number of possible outcomes, but also on the probability of each outcome. Unexpected events need a lot more questions to guess the message because they tell us more information we didn’t already know.

The entropy of a message is very important to computer scientists, because a message cannot be compressed to take up less space than its entropy.

However, once a message is compressed, a computer uses a system similar to a guessing game to retrieve the original message. A similar method is used in some computers that will guess what the user will type next. This can be useful for people who have difficulty typing, or for smaller systems like cellphones, where typing can be annoying. A good system needs an average of only two yes/no answers per character, and can be very helpful.
SUPPLIES REQUIRED:

- none

EXERCISE: TWENTY GUESSES

This is an adapted version of 20 questions. Children may ask questions to the chosen student, who may answer "yes" or "no" until the answer has been guessed. Some values the chosen student may put up for guessing include "a number between 1 and 100," "a number between 1 and 1,000,000," or "any whole number."

The number of questions is counted to find the answer is the "value" of the information.

What strategies did the children use? Which ones were the best?

Point out that it takes just 7 guesses to find a number between 1 and 100 if you halve the range each time. For example, guessing "is it less than 50?", then following it with 25, 12, 6, 3, and 2 would home in on the answer. Guessing a number from 1 to 200 would only take one more guess.

```
    no   x≥4?
    yes

  x≥2?
   no   yes
  x≥6?
   no   yes

x≥1?
   no    yes
x=0  x=1

x≥3?
   no    yes
x=2  x=3

x≥5?
   no    yes
x=4  x=5

x≥7?
   no    yes
x=6  x=7
```
Information is much easier to find in a sorted list. Telephone directories, dictionaries and book indexes all use alphabetical order, and life would be far more difficult if they didn’t. If a list of numbers (such as a list of expenses) is sorted into order, the extreme cases are easy to see because they are at the beginning and end of the list. Duplicates are also easy to find, because they end up together.

Computers spend a lot of their time sorting things into order, so computer scientists have to find fast and efficient ways of doing this, called “sorting algorithms. Here are a few examples of famous sorting algorithms:

**Selection Sort** -- one method a computer might use is called selection sort. This is how selection sort works: first, find the lightest weight in the set and put it to one side. Next, find the lightest of the weights that are left, and remove it. Repeat this until all the weights have been removed.

![Selection Sort Diagram]

**Quicksort** -- quicksort is another method of sorting that’s a lot faster than selection sort, particularly for larger lists. In fact, it’s one of the best methods known for sorting sets of data.

Quicksort works by selecting one of the objects at random, and placing it on one side of the balance scales. Then, compare each of the remaining objects with it, placing the lighter objects on the left, and the heavier objects on the right.

Then, repeat this process with each of the groups formed, keeping the comparative element in the center.

Once all the elements are in their own groups, the set’s been sorted.
**Insertion sort** works by removing each object from an unsorted group and inserting into its correct position in a new group. With each insert, the group of unsorted items shrinks, until eventually all of the objects are in the sorted group. Card players often use this method to sort a hand into order.

**Bubble sort** involves going through the list again and again, swapping any objects next to each other that are in the wrong order. The list is sorted when no swaps occur during a pass through the list. This is a very inefficient method, but easier to understand.

**Mergesort** is another method of sorting that uses the 'divide and conquer' approach. First, the list is divided at random into two lists of equal size (or nearly equal, if there are an odd number of items.) Each of the two half-size lists are sorted, then merged together by repeatedly removing the smaller of the two items at the front of the two lists.

How do you sort the smaller lists? Simple -- just use mergesort! Eventually, all the lists will be cut down into individual items, so there's no worry about knowing when to stop.
**SUPPLIES REQUIRED:**

For each group of students:
- a set of 8 containers of the same size, but different weights (e.g. milk cartons or film canisters filled with sand)
- balance scales
- student information sheets

**EXERCISE I : DISCUSSION**

Computers often have to sort lists of items into order. Brainstorm all the places where putting things into order is important. What would happen if these things were not in order?

Computers usually only compare two values at once. The next activity uses this restriction to give children an idea of what this is like.

**EXERCISE II : SORTING WEIGHTS**

Divide the students into groups. Each group will need a its own set of weights, and scales. Give the students the following instructions, then discuss the results:

- Fill each container with a different amount of sand or water, and seal tightly.
- Mix them up so that you no longer know the order of the weights.
- Find the lightest weight. What is the easiest way of doing this? *Note:* You are only allowed to use the scales to find out how heavy each container is. Only two weights can be compared at a time.
- Choose 3 weights at random and sort them into order from lightest to heaviest using only the scales. How did you do this? What is the minimum number of comparisons you can make? Why?
- Now sort all of the objects into order from lightest to heaviest.

When the students have finished, discuss which methods they used to order the weights, and why they work.
As we use computers more and more, we want them to process information as quickly as possible.

One way to increase the speed of a computer is to write programs that use fewer computational steps.

Another way to solve problems at higher speeds is to have several computers work on different parts of the same task at the same time. For example, in the six-number sorting network, although a total of 12 comparisons are used to sort the numbers, three comparisons can be performed at the same time, or “in parallel.” This means that the time required will be the same as that needed for just 5 comparison steps.

However, not all tasks can be completed faster by using parallel computation. As an analogy, imagine one person digging a ditch ten meters long. If ten people each dug one meter of the ditch, the task would be completed much faster. However, the same strategy can’t be used to dig a ditch ten meters deep - the second meter is not accessible until the first meter has been dug.

Computer scientists are still actively trying to find the best ways to break problems up so that they can be solved by computers working in parallel.
SUPPLIES REQUIRED:
- chalk
- photocopied number cards
- stopwatch

EXERCISE 1: SORTING NETWORKS

Prior to the activity, use chalk to mark out the above network on a court. Then give the students the following instructions:

- Organize yourselves into groups of six. Only one team uses the network at a time.
- Each team member takes a numbered card.
- Each member stands in a square on the left hand (IN) side of the court. Your numbers should be in jumbled order.
- You move along the lines marked, and when you reach a circle, you must wait for someone else to arrive.
- When another team member arrive in your circle, compare your cards. The person with the smaller number takes the exit to their left. If you have the higher number on your card, take the right exit.
- Are you in the right order when you get to the other end of the court? If a team makes an error, the children must start again.
- Check that you have understood the operation of a node (circle) in the network, where the smaller value goes left, and the other goes right. For example:
EXERCISE II: EXTENSION

Variations:

- When the students are familiar with the activity, use a stopwatch to time how long each team takes to get through their network.
- Use cards with larger numbers. (e.g. the three-digit photocopies.)
- Make up cards with even larger numbers, or use words and compare them alphabetically.

Extension exercises:

- What happens if the smaller number goes right instead of left, and vice versa? (The numbers will be sorted in reverse order.) What if the network is used backwards? (It won't necessarily work. Have the students find an input set where the network won't work backwards.)

- Try to design smaller or larger networks, such as the network to the right which will sort three numbers.

- Below are two different networks that will each sort four numbers. Which is faster? (The second one. Whereas the first requires the comparisons done one at a time, the second has some being performed at the same time. The first network is an example of "serial processing," while the second uses faster "parallel processing."

- Networks can also be used to find the smallest or largest of the input values. To the left is an example of a network with eight inputs, and the single output will be the minimum of the inputs. (The other values will be left at the dead end of the network.)

- What processes in everyday life can or can't be sped up using parallelism? For example, cooking a meal would be a lot slower when using only one pot, since the dishes would have to be cooked one after another.
The Muddy City: Once upon a time, there was a city that had no roads. Getting around the city was particularly difficult after rainstorms because the ground became very muddy - cars got stuck in the mud and people got their boots dirty. The mayor of the city decided that some of the streets should be paved, but didn't want to spend more money than necessary. Therefore, he specified two conditions:

1. Enough streets must be paved so that it would be possible for everyone to travel from their house to anyone else's house using only the paved roads, and
2. The paving should cost as little as possible.

Here is a map of the city. The number of paving stones between each house represents the cost of paving that route. Find the best route that connects all the houses, but uses as few counters (paving stones) as possible. How did you solve this problem?

Here is another way of presenting the houses and roads. The houses are represented by circles, the roads by lines, and the cost of paving the road is given by the number beside the line. The lengths do not have to be drawn to scale.

Computer scientists often use this sort of diagram to represent these problems. They call it a graph.
SUPPLIES REQUIRED:

- "The Muddy City" problem sheet
- counters or squares of cardboard (approx. 40 per student)

EXERCISE I : THE MUDDY CITY

This activity explores the concept of how computers are used to find the best solutions for real-life problems such as how to link power lines between houses.

Distribute the counters among the students, then give them time to find their own solutions to the Muddy City activity.

EXERCISE II : DISCUSSION

Share the solutions the students found. What strategies did they use?

One good strategy is to finding the best solution is to start with an empty map, and gradually add counters until all of the houses are linked, adding the paths in increasing order of length, but not linking houses that are already linked. Different solutions are found if you change the order in which paths of the same lengths are added. Two possible solutions are shown below:

![Diagram of Muddy City solutions]

Another strategy is to start with all of the paths paid, and remove paths you don't need, but this strategy takes more effort.

The task completed here is finding the minimal total length, or the minimal spanning tree. There are several efficient algorithms (methods) for solving minimal spanning tree problem.

There are also many other problems that can be solved using similar graphs, such as finding the shortest distance between two points, or finding the shortest route that visits all the points. The latter is also called the "travelling salesman" problem.
Routing and deadlock are problems in many networks, such as road systems and telephone or computer systems. Engineers spend a lot of time figuring out how to solve these problems -- and how to design networks that make the problem easier to solve.

Routing, congestion and deadlock can present frustrating problems in many different networks. A familiar example is rush hour traffic. Several times, the traffic in New York City has been so bad that none of the cars were able to move. Similarly, sometimes, when computers are “down” in businesses, such as banks, the problem is caused by a communication network deadlock. Designing networks so that communication in all directions is easy and efficient, and blocks are minimized is a difficult problem.

Sometimes, more than one person wants the same data at the same time. For example, in a bank, if a person's data, such as their account balance, is being updated, it is important to “lock” it so that no other person attempts to change it at the same time -- otherwise, it can cause errors. However, if this locking is interfered with by the locking another item, a data deadlock can happen.

One of the most exciting developments in computer design is that of parallel computing, where hundreds or thousands of individual processors are combined in a network to form a single powerful computer. Many problems like the Orange Game (but much larger) must be played on these networks all the time in order for these parallel computers to work.
Computers operate by following a list of instructions, called a program, that has been written to carry out a particular job. Programs are written in languages that have been specially designed to tell computers what to do, but are limited in what they can say. There are various different languages that are each suitable for writing different instructions.

But no matter what language is being used, programmers must take care to tell the computer exactly what they want to be done. Unlike people, computers will carry out instructions to the letter, even if they don’t make sense, or would be pointless.

For example, if you told a person to add numbers together, starting from 1 and counting up, until they reached -1, they would know this is impossible, and wouldn’t carry out the job. Computers, on the other hand, will add the numbers for as long as the instructions remain unchanged.

As a result, it is important that programs are well-written. A small error can cause a lot of problems. Imagine what would happen if the program for a computer in a nuclear power plant or a space shuttle launch went wrong! Errors are commonly called “bug,” in honor of a moth that was once removed (“debugged”) from an early 1940s electronic calculating machine.
SUPPLIES REQUIRED:
- example picture sheet
- pencil, paper and ruler for each student

EXERCISE I: MARCHING ORDERS

See if the students can draw the picture from these instructions:
- Draw a dot in the center of your page.
- Starting at the top left-hand corner of the page, rule a straight line through the dot, finishing at the bottom right hand corner.
- Starting at the bottom left-hand corner of the page, rule a line through the dot, finishing at the top right hand corner.
- Write your name in the triangle in the center of the left-hand side of the page.
- The result should look something like the image to the right.

EXERCISE II: IMAGE REPRODUCTION

Choose a student and give them an image (such as an image from the example sheet). The student describes the picture for the class to reproduce. The other students may ask questions to clarify the instructions. The objective is to see how quickly and accurately the exercise can be completed.

Repeat the exercise, but this time the students are not allowed to ask questions. It is best to use a simpler picture for this exercise, as the children can get lost very quickly.

Now try the exercise with the instructing student hidden behind a screen, without allowing any questions, so that the only communication is in the form of instructions. Point out that this form of communication is most like the one that computer programmers experience when writing programs. They give a set of instructions to the computer, and don’t find out the effects of the instructions until afterwards.
You may have noticed through these activities that computer science can be used by everyone to help solve everyday problems more quickly and efficiently.

However, science and technology is also used by many people on a much larger scale. Researchers are constantly using computers to make new discoveries. Doctors use advanced technology in order to better treat their patients. National leaders often look for new ways to help the citizens of their countries.

A quick survey of current events can reveal dozens of situations where science, technology, engineering and math would help.

Look at the front page of a few news services and see how many times science and technology are involved.
Supplies Required:

- none

Exercise 1: Current Events

In addition to thinking computationally, students need to be aware of how science and technology impact their day-to-day lives. A quick survey of current events, with an eye towards STEM, results in dozens of opportunities to put science, technology, engineering and math to use. Look at the front page of any news services and see how many times STEM is involved.

Some new sites that may be helpful:

- ABC - http://abcnews.go.com
- MSNBC - http://www.msnbc.msn.com
- Science Magazine - http://www.sciencemag.org
- Nature - http://www.nature.org
- National Geographic - http://www.nationalgeographic.com
For Scientists

★ Supercomputers allow scientists to model processes that are too large, costly, or complicated to investigate in other ways.

★ Computer modeling and simulations are a new way of doing science, along with experimentation, to test hypotheses.

★ Larger and more complex computers are required as models and simulations continue to grow.

★ Even the fastest computers today are not powerful enough to model many of the systems scientists want to study.

For you

★ Develop a simulation using a molecular modeling tool to explore why water expands when it freezes.

★ Compare different types of IRAs using financial analysis tools to determine which investment would earn more.

★ Develop a model of the carbon cycle to study the effects of greenhouse gases in the atmosphere.
EDUCATION

- Bioinformatics
- Education
- Electrical engineering
- Software engineering
- Computer modeling
- Computational biology
- Computational chemistry
- Computer systems management
- Neuroscience
- Pathology
- Earth sciences
- Humanities & Arts
- Data transfer and storage

CAREER PATHWAYS

- Developing the next generation of Internet technologies
- Producing storm simulation data for more accurate weather predictions
- Developing data analysis tools and distributing resources for research
- Simulating biological systems (proteins, DNA, membranes, etc.) for research
- Designing and managing high performance computing systems and massive data storage systems

COLLEGE INTERNSHIPS

The Pittsburgh Supercomputing Center brings in approximately 25 undergraduate interns each year. Interns work in teams actively developing tools to improve the high performance computing environment, as well as computational tools used within PSC. All receive valuable experience and contribute significantly to a high performance computing trained workforce. Some of the best have become PSC employees!

Internships may last for a summer, or a full academic year. We will also support the student academically if s/he wants to do the internship for academic credit.

Recent accomplishments:

- A 3D monitor that displays job and environmental information for our Cray XT3 machine.
- A remote reset switch for geographically distributed, but centrally administered servers (essential to running a dark machine room)
- Developing a code profiling tool for predicting application performance on future architectures and optimizing user codes.

Special effort is made to recruit interns from underrepresented groups.
For more information, contact Laura McGinnis at lfm@psc.edu.
Carnegie Mellon

The materials presented in this document are included with the permission of the contributing authors as a mechanism to disseminate scholarly/technical information on a non-commercial basis. Copyright and all rights therein are maintained by the authors. It is understood that all individuals copying this information will adhere to the terms/constraints invoked by each author’s copyright.

This material may not be copied for commercial redistribution, republication, or dissemination without the explicit permission of Carnegie Mellon University and the authors.