Making a Computer Work: Searching[2]

Introduction

"Computers store a lot of information, and they need to be able to sift through it quickly. One of the biggest search problems in the world is faced by Internet search engines, which must search billions of web pages in a fraction of a second. "

"Computers can process information very quickly, and you might think that to find something they should just start at the beginning of their storage and keep looking until the desired information is found." This is what we do in Activity Linear Search. "But this method is very slow—even for computers. For example, suppose a supermarket has 10,000 different products on its shelves. When a bar code is scanned at a checkout, the computer must look through up to 10,000 numbers to find the product name and price. Even if it takes only one thousandth of a second to check each code, ten seconds would be needed to go through the whole list. Imagine how long it would take to check out the groceries for a family!

A better strategy is *binary search*. In this method, the numbers are sorted into order. Checking the middle item of the list will identify which half the search key is in. The process is repeated until the item is found. Returning to the supermarket example, the 10,000 items can now be searched with fourteen probes, which might take two hundredths of a second—hardly noticeable. "

[2] Adapted from CS Unplugged Activity "Searching Algorithms". Online: http://csunplugged.org/searching-algorithms/

Activity: Linear Search

Each student is given a card with a number on it (in random order). The students should keep the number hidden.

One student is given sweets and a task: **Find a given number**. To look at a card from another student, they have to "pay" that student with a sweet. The objective is to keep the largest amount of sweets possible.

This can be repeated a few times with different students.

Activity: Binary Search

Students are given a card with a number on it again, but now numbers will be ordered (for example in ascending order).

Again, one student is given sweets and a task: **Find a given number**. He still has to "pay" to look at a card from another student.

By using one payment that eliminates half the students, the right number can be found more efficiently. This is called *binary search*.

Making a Computer Work: Sorting[3]

Introduction

"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. Some of the slower methods such as insertion sort, selection sort and bubble sort can be useful in special situations, but the fast ones such as quicksort are usually used.

Quicksort uses a concept called recursion. This means you keep dividing a list into smaller parts, and then performing the same kind of sort on each of the parts. This particular approach is called divide and conquer. The list is divided repeatedly until it is small enough to conquer. For quicksort, the lists are divided until they contain only one item. It is trivial to sort one item into order! Although this seems very involved, in practice it is dramatically faster than other methods."

Activity: Selection Sort

Organise students in groups. Each group has a set of 8 similar containers (e.g. soda cans with sand or water), and 20 sweets. There is one person that can "weight" the containers. This person does not provide the weight of the containers; he/she only compares two containers and tells which one is the heaviest. This "service" needs to be "paid" with one candy per comparison.

Each group has one task: **Order the containers by weight (lightest to heaviest).**

To make a selection sort, proceed as follows. 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.

How many candies you have left? Can you think of ways to do this that require fewer comparisons?

---

[3] Adapted from CS Unplugged Activity "Sorting Algorithms". Online: http://csunplugged.org/sorting-algorithms/

ACTIVITY: QUICK SORT

Organise students in groups as in the previous activity.

The task is again: **Order the containers by weight (lightest to heaviest).**

To make a quick sort, proceed as follows. Choose one of the containers at random, and place it on the middle of the table. Now compare each of the remaining objects with it. Put those that are lighter on the left of the table, and the heavier ones on the right. (By chance you may end up with many more objects on one side than on the other.)

Repeat this procedure (choosing a random container and dividing them between heavier and lighter) to both groups. Keep repeating this procedure on the remaining groups until no group has more than one object in it. Once all the groups have been divided down to single objects, the objects will be in order from lightest to heaviest.

How many candies you have left? Think of the different groups that could be formed according to the random container selected.