Many Campers Sort Piles — Instructor: Padraic Bartlett

Lecture 5: A Field Guide to Sorting Algorithms

Week 4 — *Mathcamp 2012*

Throughout this class, we've talked about a large number of sorting algorithms. This guide is a quick reminder and restatement of the sorting algoritms we've discussed in class so far!

1 Bogosort

Bogosort is the *best* sorting algorithm.

0. Input: a list of integers.
1. Given this list of integers, we first check if the list is sorted. If it is, then we stop! Our list is clearly sorted.
2. Otherwise, **randomly permute** the elements in this list, and go back to (1).

Basically, bogosort is 52-card-pickup turned into a sorting algorithm.

2 Bubblesort

Bubblesort is a relatively classical and beautiful sorting algorithm: Take as input some list $L = (l_1, \dots l_n)$ of objects, along with some operation $\leq$ that can compare any two of these elements. Perform the following algorithm:

1. Create a integer variable loc and a boolean variable $didSwap$.
2. Set the variable loc to 1, and $didSwap$ to $false$.
3. While the variable loc is $< n$, perform the following steps:
   (a) If $l_{loc} > l_{loc+1}$, swap the two elements l_{loc}, l_{loc+1} in our list and set $didSwap$ to true.
   (b) Regardless of whether you swapped elements or not, add 1 to the variable loc, and go to 3.
4. If $didSwap$ is $true$, then go to 2.
5. Otherwise, if $didSwap$ is $false$, we went through our entire list and never found any pair of consecutive elements such that the second was larger than the first. Therefore, our list is sorted! Output our list.

3 Beadsort

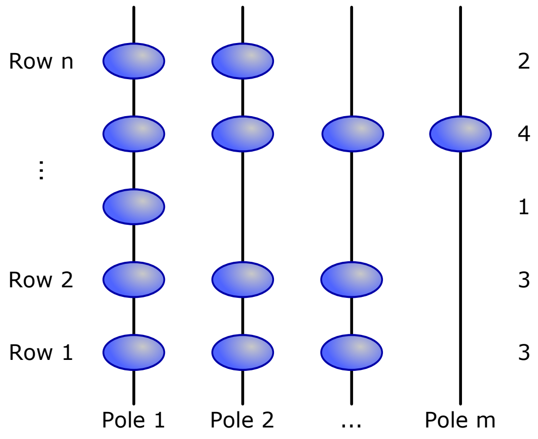
Beadsort was a particularly beautiful algorithm we discussed on the first day:

Algorithm. Take as input some list $L = (l_1, \dots l_n)$ of positive integers; as well, have on hand $m = \sum_{i=1}^{n} l_i$ many beads, and $\max_{i=1}^{n} l_i$ many poles on which to place these beads.

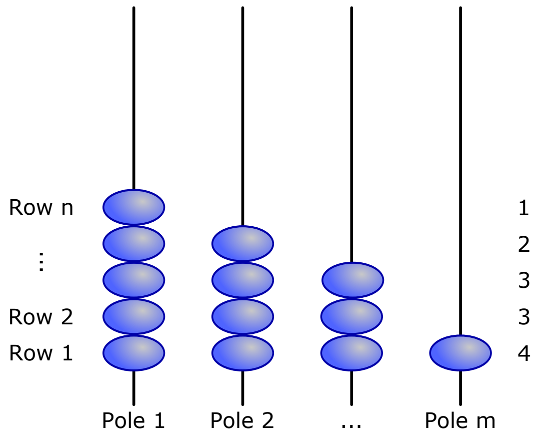
1. Label our poles $1 \dots m$.
2. For each element l_i in our list, place one bead on each of the poles $1, 2 \dots l_i$.
3. Let gravity occur.
4. Starting from height n and working your way down, read off the number of beads at that given height. Write down this number as l_k.

This algorithm is perhaps best illustrated with a pair of pictures:

Setup:



Result:



4 Quicksort

Quicksort was perhaps our first "serious" sorting algorithm. It goes as follows: given a list $L = (l_1, \dots l_n)$,

- Pick an element l_k out of our list. Call this a **pivot** element.
- Take the rest of our list, and split it into two lists: the list $L_<$ of elements less than our pivot, and the list $L_\geq$ of elements that are greater than or equal to our pivot.
- Quicksort the two lists $L_<$ and $L_\geq$.

5 Mergesort

Mergesort was our second "serious" sorting algorithm, and was beautifully guaranteed to run in $O(n \log(n))$ time:

- Take our list, and split it into two lists.
- Recursively mergesort each of these lists.
- Now, "merge" these two sorted lists, by repeatedly looking at the smallest elemenet in each list and placing the smaller one into our final list.

6 Stoogesort

Consider the following algorithm (Stoogesort[1]!) from the second HW for sorting a list: Take as input a list $L = (l_1, \dots l_n)$.

- If your list contains one or two elements, sort it by just looking at the list.
- Otherwise, the list contains ≥ 3 elements. Let $M = \lceil 2n/3 \rceil$.
- Stoogesort the list $(l_1, \dots l_m)$.
- Stoogesort the list $(l_{n-m}, \dots l_n)$.
- Stoogesort the list $(l_1, \dots l_m)$.

7 Bucketsort

Bucket sort, or bin sort, is a sorting algorithm that works by partitioning an array into a number of buckets! Specifically, do the following: let n, m be the minimum/ maximum values that numbers in your list attain, and create some number of buckets that partition the integers from n to m.

- Go over the array, putting each element into its bucket.

[1]Named after the comedy routines of the Three Stooges; specifically, the ones where each stooge hits the other two.

- Sort each bucket (either recursively, by bucket-sorting each bucket, or by applying another algorithm.)
- Gather all of your sorted data back together and turn it into a sorting of your original list.

8 Sleepsort

Sleepsort is perhaps most famous for being the only sorting algorithm discovered by 4chan. It runs in "linear" time, for very stupidly defined values of linear, and we define it here:

0. Input: a list $\{l_1, \dots l_n\}$ of n distinct integers. Also, someone with a stopwatch, and another person with a pen.
1. Start your stopwatch.
2. Every time the number of seconds is equal to a number on the list, the person with the stopwatch should shout out the number, and the person with a pen should write it down.
3. After a number of seconds equal to the size of the largest element in your list, you've written down all of the numbers in order! Win.

Properties of sleepsort:

- Sleepsort's name comes from the **sleep** command-line program, which on input n creates a process that waits n seconds before doing anything else. The following script is an implementation of sleepsort:

```
#!/bin/bash
function f() {
    sleep "$1"
    echo "$1"
}
while [ -n "$1" ]
do
    f "$1" &
    shift
done
wait
```