Sorting

Introduction

- Why is it important
- Where to use it

Topics of Discussion

- Simpler methods (faster coding)
  - Bubble
  - Insertion
  - Selection
- Faster methods (faster execution)
  - Merge
  - Bucket
  - Radix

Bubble sorting

- For i=1 to n-1
  {
  for j=1 to i
  if value[i]>value[i+1]
  switch value[i] and value[i+1];
  }

Insertion sorting

- For i=1 to n
  {
  temp=value[i];
  find where value[i] should be in the already sorted values 1 to i-1, e.g. position k;
  shift all sorted values after k one place to the right;
  value[k]=temp;
  }

Selection sorting

- For i=1 to n
  {
    find the biggest value between i and n and switch it with the value in position i;
  }

Merge sorting

- Merge two sorted arrays into a new array
- e.g.
  - Step 1: new: empty
    arr1: 11, 23, 42
    arr2: 9, 25
  - Step 2: new: 9
    arr1: 11, 23, 42
    arr2: 25
  - Step 3: new: 9, 11
    arr1: 23, 42
    arr2: 25
  - etc.
- An unsorted array of length n can be split

Merge sorting (cont.)

- An unsorted array of length n can be split into n sorted arrays of length 1 (an array with only one element is always sorted)
- Recursively merge those n arrays to end with one sorted array

Bucket sorting

- E.g. sort n integers each with a value between 1 and m
  - Create an array arr[] with size m
  - Pass through the original array and every time the number k occurs, increment arr[k]
  - Or use a linked list for each value
- Not a very good option when m is very big

Radix sorting

- Better for sorting bigger integers
- Bucker sort using only one digit at a time, starting with the least significant digit: the last bucket sort alters the final order the most so it should be with the most significant digit.
  - Use a linked list for each value
  - After each bucket sort concatenate the lists
- Optimization: use a base larger than 10 or a power of 2

Sorting floating point numbers

- Floating point number x = a*10^b
- First sort by a and then by b
- Any base can be used instead of 10

Summary

- **Bubble**: loop through and switch places
- **Insertion**: find correct place and insert there
- **Selection**: select the next biggest number and place it
- **Merge**: merge sorted arrays or lists (recursively if necessary)
- **Bucket**: create buckets (the values) and place each item in the right bucket
- **Radix**: repeatedly bucket-sort using the different digits starting from the least significant digit