Animation Demos

http://www-cse.uta.edu/~holder/courses/cse2320/lectures/applets/sort1/heapsort.html

http://cg.scs.carleton.ca/~morin/misc/sortalg/
Sorting Algorithms

- Selection Sort
- Insertion Sort
- Bubble Sort
- Shaker Sort
- Shell Sort
- Merge Sort
- Heap Sort
- Quick Sort

- Bucket & Radix Sort
- Counting Sort
QuickSort(A, p, r)
  if (p < r) then
    q = Partition(A, p, r)
    QuickSort(A, p, q)
    QuickSort(A, q+1, r)

Partition(A, p, r)
  x = A[r]
  i = p-1
  for j = p to r-1 do
    if (A[j] <= x) then
      i++
      SWAP(A[i], A[j])
  SWAP(A[i+1], A[r])
  return i+1
Figure 8.5
Shellsort after each pass if the increment sequence is \{1, 3, 5\}

<table>
<thead>
<tr>
<th>Original</th>
<th>81</th>
<th>94</th>
<th>11</th>
<th>96</th>
<th>12</th>
<th>35</th>
<th>17</th>
<th>95</th>
<th>28</th>
<th>58</th>
<th>41</th>
<th>75</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>After 5-sort</td>
<td>35</td>
<td>17</td>
<td>11</td>
<td>28</td>
<td>12</td>
<td>41</td>
<td>75</td>
<td>15</td>
<td>96</td>
<td>58</td>
<td>81</td>
<td>94</td>
<td>95</td>
</tr>
<tr>
<td>After 3-sort</td>
<td>28</td>
<td>12</td>
<td>11</td>
<td>35</td>
<td>15</td>
<td>41</td>
<td>58</td>
<td>17</td>
<td>94</td>
<td>75</td>
<td>81</td>
<td>96</td>
<td>95</td>
</tr>
<tr>
<td>After 1-sort</td>
<td>11</td>
<td>12</td>
<td>15</td>
<td>17</td>
<td>28</td>
<td>35</td>
<td>41</td>
<td>58</td>
<td>75</td>
<td>81</td>
<td>94</td>
<td>95</td>
<td>96</td>
</tr>
</tbody>
</table>
public static void shellsort( Comparable [ ] a )
{
    for( int gap = a.length / 2; gap > 0;
        gap = gap == 2 ? 1 : (int) ( gap / 2.2 ) )
    for( int i = gap; i < a.length; i++ )
    {
        Comparable tmp = a[ i ];
        int j = i;

        for( ; j >= gap && tmp.compareTo( a[ j - gap ] ) < 0; j -= gap )
            a[ j ] = a[ j - gap ];
        a[ j ] = tmp;
    }
}
Sorting Algorithms

- Selection Sort
- Insertion Sort
- Bubble Sort
- Shaker Sort
- Shell Sort
- Merge Sort
- Heap Sort
- Quick Sort
- Bucket & Radix Sort
- Counting Sort
Bucket Sort

- N values in the range \([a..a+m-1]\)
- For e.g., sort a list of 50 scores in the range \([0..9]\).
- Algorithm
  - Make m buckets \([a..a+m-1]\)
  - As you read elements throw into appropriate bucket
  - Output contents of buckets \([0..m]\) in that order
- Time \(O(N+m)\)
Stable Sort

- A sort is stable if equal elements appear in the same order in both the input and the output.
- Which sorts are stable? Homework!
Radix Sort

Algorithm

\[\text{for } i = 1 \text{ to } d \text{ do}\]

\text{sort} array A on digit i using a stable sort algorithm

Time Complexity: \(O((n+k)d)\)