Shellsort and Shellsort networks

Ancient results

Old results

Average-case analysis

Variants of Shellsort

New ideas
Shellsort

```c
shellsort(itemType a[], int l, int r)
{
    int incs[16] =
    { 1391376, 463792, 198768, 86961, 33936,
       13776, 4592, 1968, 861, 336, 112, 48,
       21, 7, 3, 1 };
    int i, j, h, v;

    for ( k = 0; k < 16; k++ )
        for (h = incs[k], i = l+h; i <= r; i++)
            { v = a[i]; j = i;
              while (j > h && a[j-h] > v)
                  { a[j] = a[j-h]; j -= h; }
              a[j] = v;
            }
}
```

Running time depends on increment sequence

Solved problem:
* running time is

Open problems:
* "best" increment sequences for practical N
* average-case analysis for any interesting sequence
* N log N variants
* variants corresponding to log N depth networks
UPPER BOUND

Use following increments

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>3</td>
<td>6</td>
<td>12</td>
<td>24</td>
<td>48</td>
<td>96</td>
<td>192</td>
<td>384</td>
</tr>
<tr>
<td>.</td>
<td>9</td>
<td>18</td>
<td>36</td>
<td>72</td>
<td>144</td>
<td>288</td>
<td>576</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>27</td>
<td>54</td>
<td>108</td>
<td>216</td>
<td>432</td>
<td>864</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>81</td>
<td>162</td>
<td>324</td>
<td>648</td>
<td>1296</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>243</td>
<td>486</td>
<td>962</td>
<td>1924</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>729</td>
<td>1458</td>
<td>2916</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>2187</td>
<td>4374</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>6561</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total running time is

Applies to networks

Too slow in practice

LOWER BOUND

If increment sequence is "almost geometric"
then total running time must be
Use the following increments

1   8   23   77   281   1073   4193   16577   . . .

Increment sequence not "almost geometric"

Connection to "Frobenius problem"

Smaller of two bounds

first bound

second bound

Use first bound for small increments

Use second bound for large increments

Total running time is
Frobenius Problem

A country wishes to issue \( k \) different stamps

* Number of values that cannot be achieved?
* Largest value that cannot be achieved?

Examples

Two stamps, relatively prime (Curran-Sharp, 1884)

Three stamps (Selmer, 1977)
### Chazelle Upper Bound

Generalize Pratt "network" construction

**Example**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
<td>56</td>
<td>392</td>
<td>19208</td>
<td>134456</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>49</td>
<td>343</td>
<td>2744</td>
<td>19208</td>
<td>134456</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>343</td>
<td>2401</td>
<td>3136</td>
<td>21952</td>
<td>153664</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2401</td>
<td>16807</td>
<td>3584</td>
<td>25088</td>
<td>175616</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16807</td>
<td>117649</td>
<td>4096</td>
<td>28672</td>
<td>200704</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>117649</td>
<td>262144</td>
<td>32768</td>
<td>229376</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total running time is

Choose parameter optimally

(restrict to logarithmic number of passes)

Too slow in practice
Start with a "basis" of relatively prime numbers

\[
1 \quad 3 \quad 7 \quad 16 \quad 41
\]

Build a sequence with every number the product of a basis number and a number earlier in the sequence

\[
\begin{align*}
1 & \quad 1 \times 3 & \quad 1 \times 3 \times 7 & \quad 1 \times 3 \times 7 \times 16 & \quad 1 \times 3 \times 7 \times 16 \times 41 \\
. & \quad 1 \times 7 & \quad 1 \times 3 \times 16 & \quad 1 \times 3 \times 7 \times 41 & \quad 1 \times 3 \times 7 \times 16 \times 101 \\
. & \quad 1 \times 7 \times 16 & \quad 1 \times 3 \times 16 \times 41 & \quad 1 \times 3 \times 7 \times 41 \times 101 \\
. & \quad 1 \times 7 \times 16 \times 41 & \quad 1 \times 3 \times 16 \times 41 \times 101 \\
. & \quad 1 \times 7 \times 16 \times 41 \times 101
\end{align*}
\]

\[
\begin{align*}
1 & \quad 3 & \quad 21 & \quad 336 & \quad 13776 \\
. & \quad 7 & \quad 48 & \quad 861 & \quad 33936 \\
. & \quad 112 & \quad 1968 & \quad 86961 \\
. & \quad 4592 & \quad 198768 \\
. & \quad 463792
\end{align*}
\]

Asymptotically optimal (same as Chazelle)

Fast in practice
Using $M$ increments on a file of size $N$ requires at least

comparisons in the worst case, for some $c > 0$.

Applies to any algorithm that

* uses a number of passes
  compare-exchanging items
  at a fixed increment
* does at least $c$ comparisions on each pass
* does not disturb $k$-ordering once achieved
Complexity "gap"

<table>
<thead>
<tr>
<th></th>
<th>1000</th>
<th>10000</th>
<th>100000</th>
<th>1000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>thousand</td>
<td>10</td>
<td>3</td>
<td>9</td>
<td>78</td>
</tr>
<tr>
<td>million</td>
<td>20</td>
<td>4</td>
<td>22</td>
<td>482</td>
</tr>
<tr>
<td>billion</td>
<td>30</td>
<td>6</td>
<td>43</td>
<td>1933</td>
</tr>
<tr>
<td>trillion</td>
<td>40</td>
<td>9</td>
<td>78</td>
<td>6233</td>
</tr>
</tbody>
</table>

**UPPER BOUND**

pass(es):

total cost:

**LOWER BOUND**

pass(es):

total cost:

**AVERAGE CASE**

No results for any interesting sequences
Simulations show
   average case close to worst case
   for sequences designed to worst case
Average case (two or three increments)

Analysis of \((h, 1)\) Shellsort (Knuth)

Analysis of \((h, k, 1)\) Shellsort (Yao)

Asymptotic result for three increments?
Shakersort (Incerpi, Sedgewick, 1984)

Shellsort "network"
  Do one "cocktail shaker" pass
    (not full insertion sort)
    for each increment
  Choose increments close to
  Always seems to sort (!)

Poonen’s bound applies; can’t always sort

Can serve as basis for probabilistic
  sorting network with N log N comparators

Variants
  try more sophisticated increment sequences
  do multiple shakes for each increment
  add 1-shakes at end if necessary

DISADVANTAGE
  network is "depth" N
Bricksort (Sedgewick, Lemke, 1995)

Shellsort "network"
  Do one "brick" pass
    (not full shaker pass)
    for each increment
  Choose increments close to
  Always seems to sort (!!)

Poonen’s bound applies?

Can serve as basis for probabilistic
  sorting network of "depth" log N

Variants
  try more sophisticated increment sequences
  do multiple brick passes for each increment
  add 1-passes at end if necessary

Average-case analysis??
Analysis of Bricksort