Adam7

Submit Solution

The Adam7 algorithm is a technique for progressive loading of images used in the PNG format. The technique consists of breaking the image into 7 sub-parts, applying the following 8×8 pattern on all the pixels of the image:

```
1 6 4 6 2 6 4 6
7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6
7 7 7 7 7 7 7 7
3 6 4 6 3 6 4 6
7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6
7 7 7 7 7 7 7 7
```

In this way, it is possible to reconstruct (roughly) the image using only the values up to a certain level, thus displaying a preview as the loading takes place, or to have a reduced version of the image by downloading only the level of detail one is interested in.

Write a command-line program that accepts the following options:

```
multires c <input file .PGM> <output file .MLT>

multires d <input file .MLT> <output prefix>
```

When the “c” option is specified, the program opens the .PGM file and creates the file in .MLT format structured as follows:

Field	**Size**	**Description**
Magic Number	8 bytes	"MULTIRES"
Width	32 bits unsigned integer little endian	Number of columns
Height	32 bits unsigned integer little endian	Number of rows
Level 1		All pixels (row wise) that correspond to value 1 in the Adam7 pattern
Level 2		All pixels (row wise) that correspond to value 2 in the Adam7 pattern
…	…	…
Level 7		All pixels (row wise) that correspond to value 7 in the Adam7 pattern

When the “d” option is specified, the program opens the .MLT file and creates seven .PGM files corresponding to the reconstructed image using only level 1, then levels 1 and 2, then levels 1, 2 and 3 etc. If the prefix were `rec`, the program would output `rec_1.pgm`, `rec_2.pgm` and so on up to `rec_7.pgm`.

For the reconstruction, you can use a technique without interpolation, which consists in copying the lower level pixels into higher levels ones accordingly to the following tables (within < and > the value actually available for the block):

• if only level 1 is available, the entire block must be filled with that value:

```
<a> a a a a a a a
a a a a a a a a
a a a a a a a a
a a a a a a a a
a a a a a a a a
a a a a a a a a
a a a a a a a a
a a a a a a a a
```

• if level 2 is available, the first four columns of each block are filled with the value of pixel 1, while the other four with that of pixel 2:

```
<a> a   a   a   <b> b   b   b
a   a   a   a   b   b   b   b
a   a   a   a   b   b   b   b
a   a   a   a   b   b   b   b
a   a   a   a   b   b   b   b
a   a   a   a   b   b   b   b
a   a   a   a   b   b   b   b
a   a   a   a   b   b   b   b
```

• if level 3 is available, the bottom two 4×4 blocks will be filled respectively with the value of your top left pixel:

```
<a> a   a   a   <b> b   b   b
a   a   a   a   b   b   b   b
a   a   a   a   b   b   b   b
a   a   a   a   b   b   b   b
<c> c   c   c   <d> d   d   d
c   c   c   c   d   d   d   d
c   c   c   c   d   d   d   d
c   c   c   c   d   d   d   d
```

• and so on for levels 4, 5, 6 and 7 (unspecified, because at that level the pixels are all defined):

```
<a> a   <e> e   <b> b   <f> f       <a> a   <e> e   <b> b   <f> f
a   a   e   e   b   b   f   f       a   a   e   e   b   b   f   f
a   a   e   e   b   b   f   f       <i> i   <j> j   <k> k   <l> l
a   a   e   e   b   b   f   f       i   i   j   j   k   k   l   l
<c> c   <g> g   <d> d   <h> h       <c> c   <g> g   <d> d   <h> h
c   c   g   g   d   d   h   h       c   c   g   g   d   d   h   h
c   c   g   g   d   d   h   h       <m> m   <n> n   <o> o   <p> p
c   c   g   g   d   d   h   h       m   m   n   n   o   o   p   p

<a> <q> <e> <r> <b> <s> <f> <t>     …   …   …   …   …   …   …   …
 a   q   e   r   b   s   f   t      …   …   …   …   …   …   …   …
<i> <u> <j> <v> <k> <w> <l> <x>     …   …   …   …   …   …   …   …
 i   u   j   v   k   w   l   x      …   …   …   …   …   …   …   …
<c> <y> <g> <z> <d> <1> <h> <2>     …   …   …   …   …   …   …   …
 c   y   g   z   d   l   h   2      …   …   …   …   …   …   …   …
<m> <3> <n> <4> <o> <5> <p> <6>     …   …   …   …   …   …   …   …
 m   3   n   4   o   5   p   6      …   …   …   …   …   …   …   …
```

When preparing the solution, first deal with the `c` option, then the `d` option by decoding only the complete `rec_7.pgm` version and, finally, handle the other levels that require you to replicate the decoded pixels on their neighbors.

There is no need to handle images with width or height that are not multiple of eight in special ways. For each pixel in the original image, you can compute its level (1 to 7) and then you know how to handle it. Let’ s assume you have a 3×4 image (values are in hexadecimal):

```
10  20  30
40  50  60
70  80  90
A0  B0  C0
```

The levels are:

```
1   6   4
7   7   7
5   6   5
7   7   7
```

The pixels will be organized in row major order, and the final file will be composed by these parts:

```
1. Magic Number: MULTIRES
2. Width: 3
3. Height: 4
4. Level 1: (1 pixel) 10
5. Level 2: (0 pixels)
6. Level 3: (0 pixels)
7. Level 4: (1 pixel) 30
8. Level 5: (2 pixels) 70 90
9. Level 6: (2 pixels) 20 80
10. Level 7: (6 pixels) 40 50 60 A0 B0 C0
```

In hexadecimal (editor view):

```
1. Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
2. 00000000  4D 55 4C 54 49 52 45 53 03 00 00 00 04 00 00 00  MULTIRES........
3. 00000010  10 30 70 90 20 80 40 50 60 A0 B0 C0              .0p. €@P` °À
```