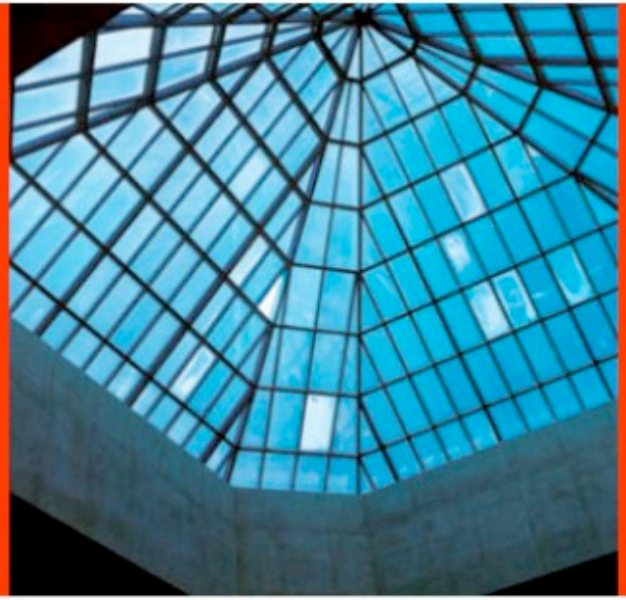

ORACLE®



ORACLE

How to Think about Parallel Programming—Not!

Guy L. Steele Jr.
Sun Labs, Oracle

The IBM 1130

- 16-bit words
- 1 32-bit register
- 4096-word memory

http://ed-thelen.org/comp-hist/vs-ibm-1130.jpg

IBM 1130 Console

- keyboard
- console printer
- 16 data entry switches

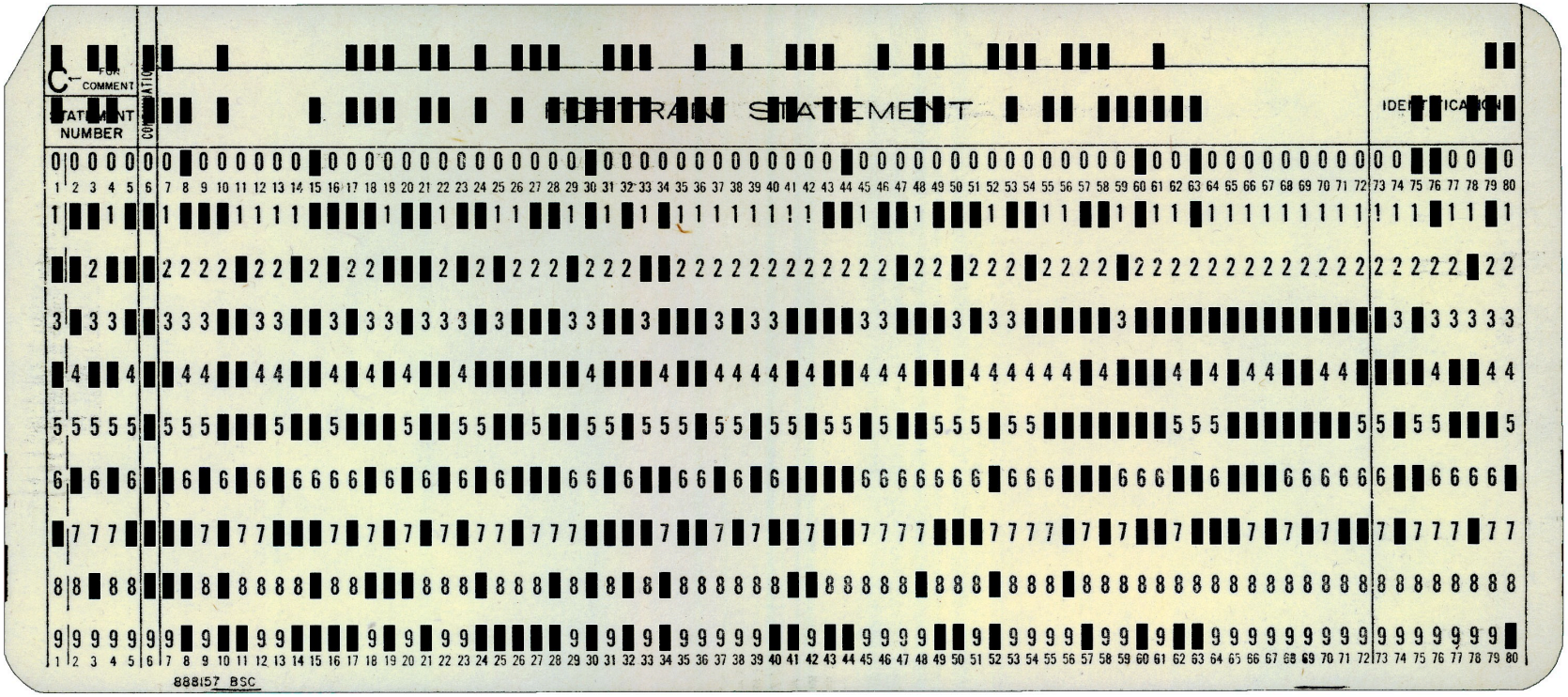
Photo by Bob Rosenbloom; used with permission.

IBM 1442 Card Read Punch

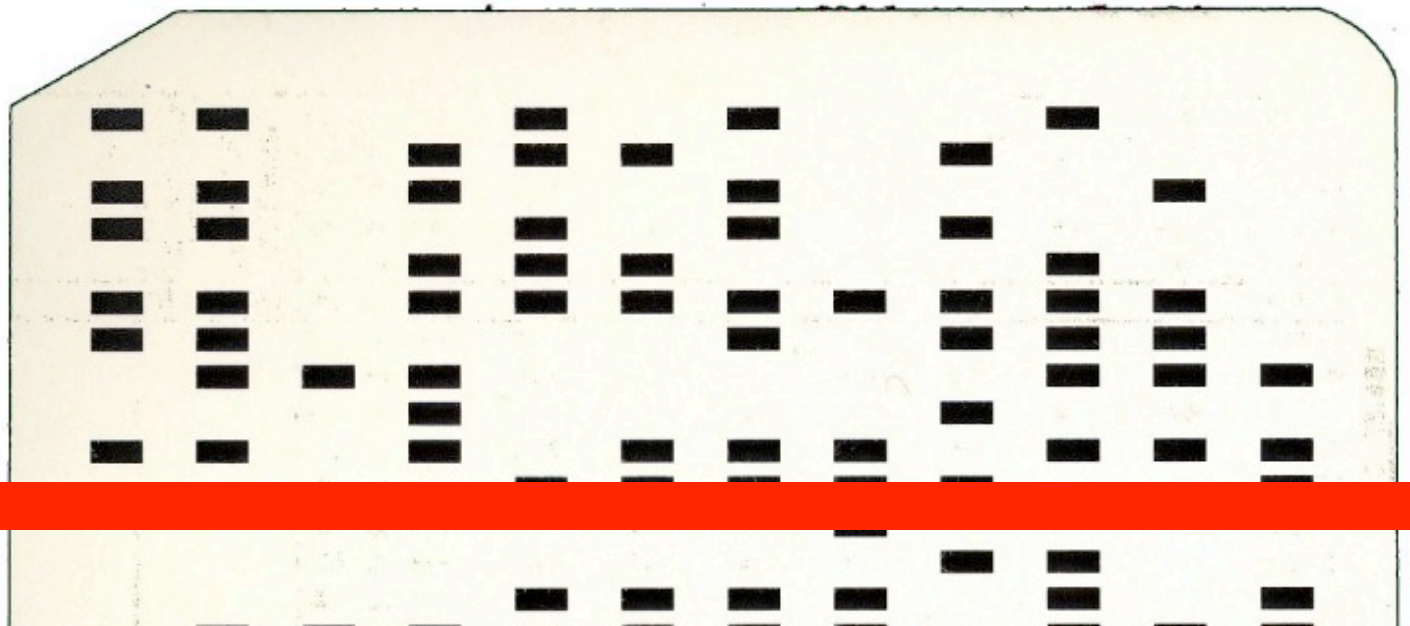
- Reads and punches 80-column cards
- Program load mode

Photo by Mike Ross

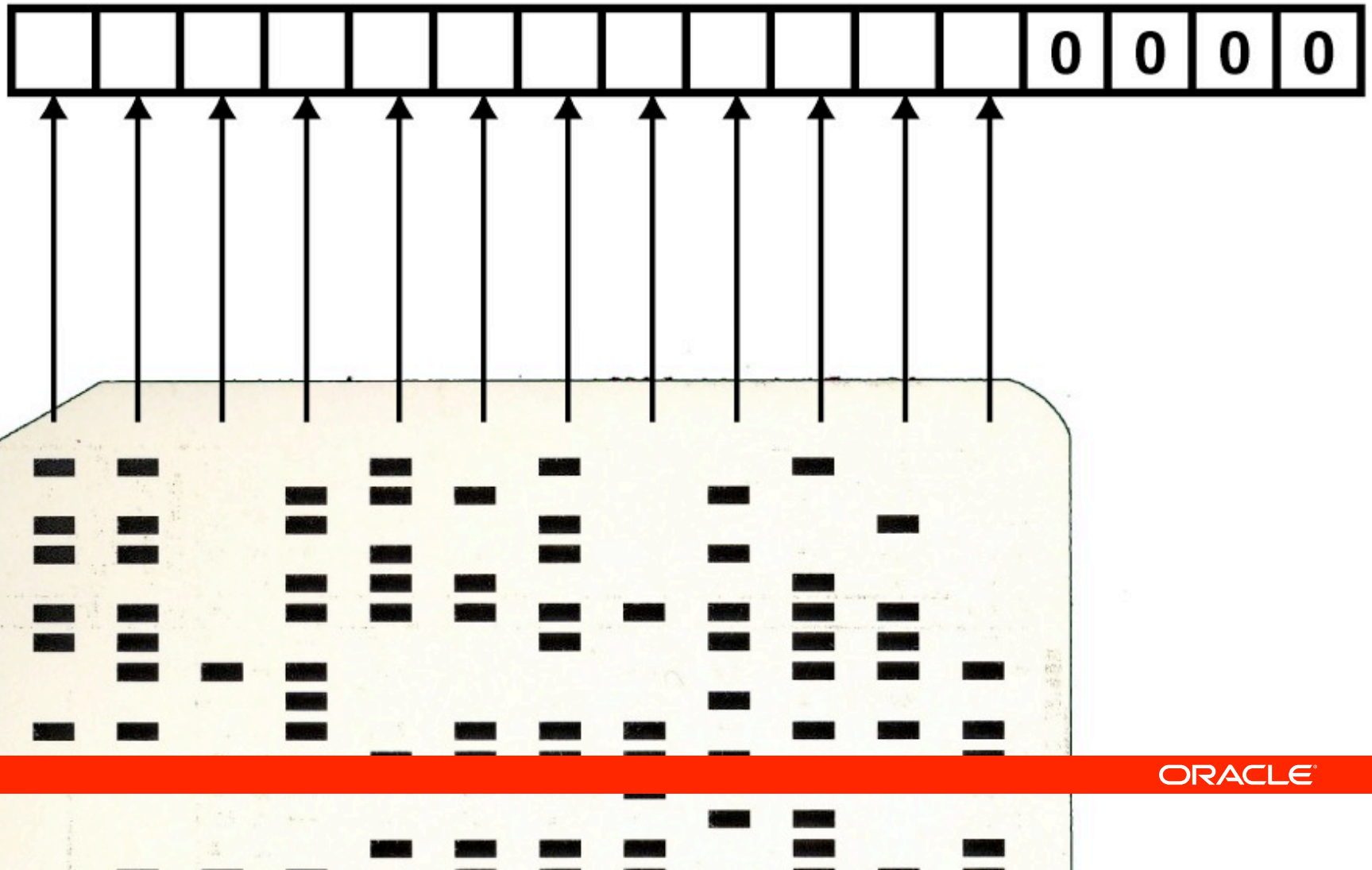
My Ugly Program



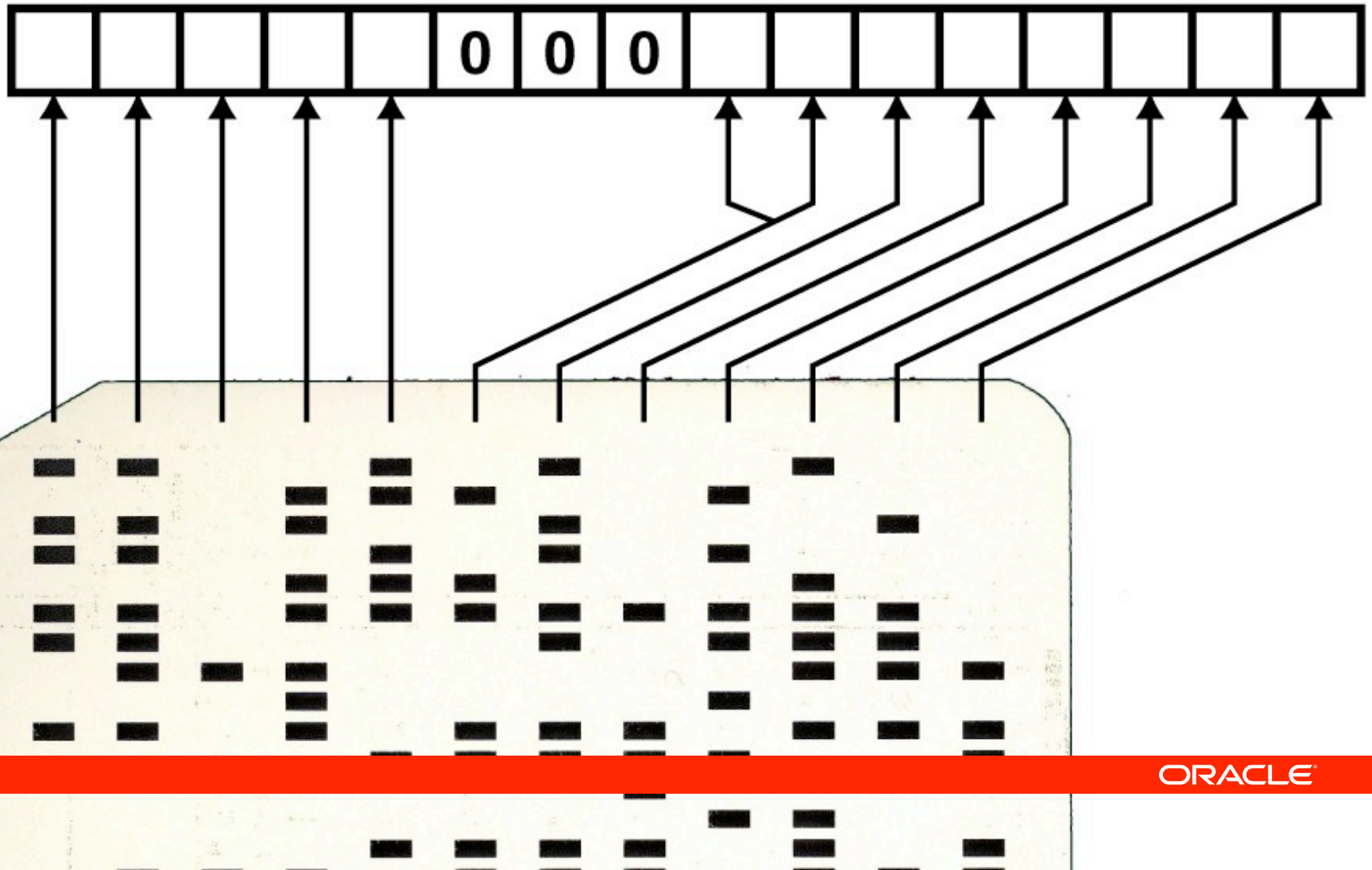
Flip the Card Over



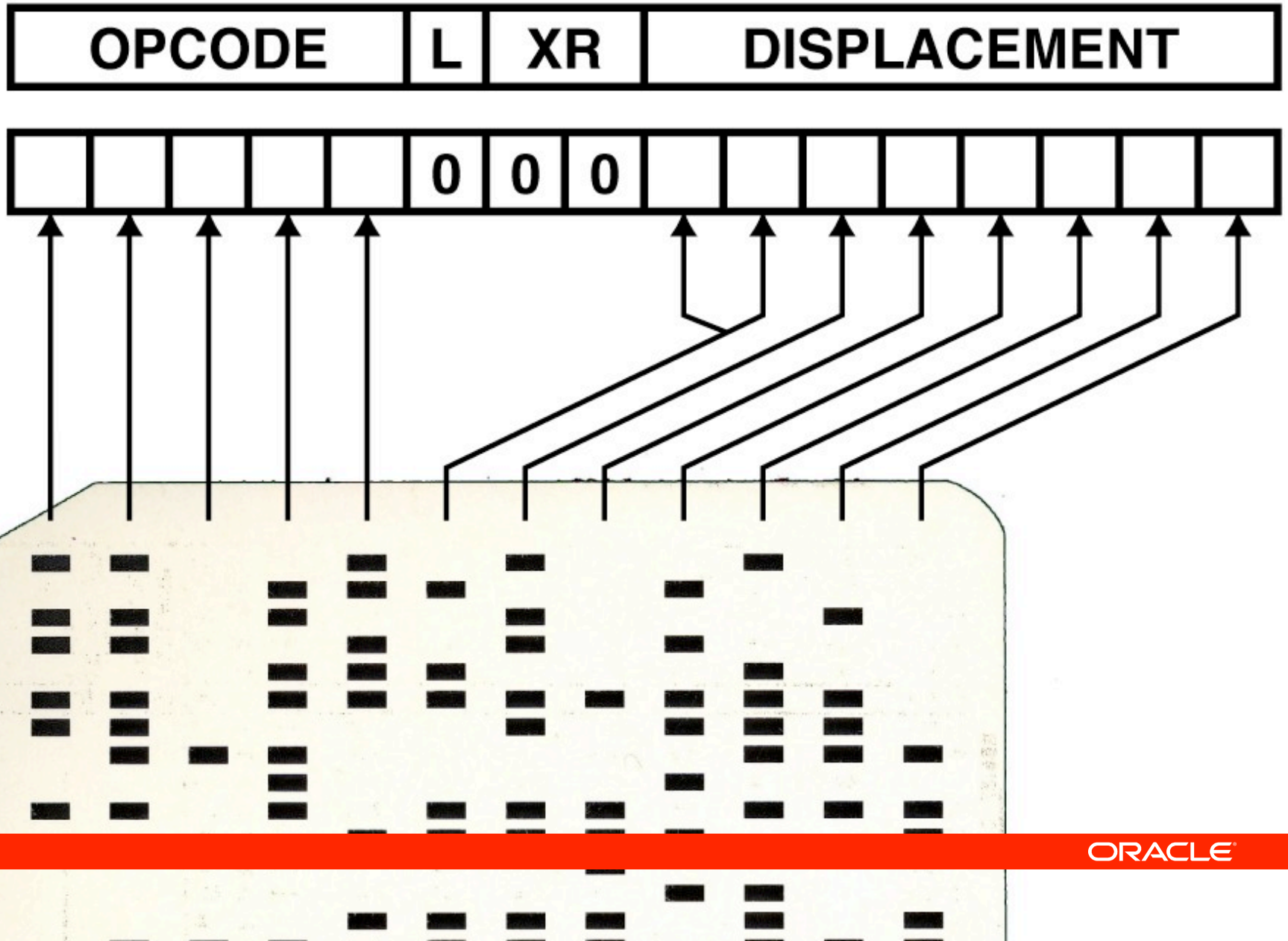
Normal Card Data Input



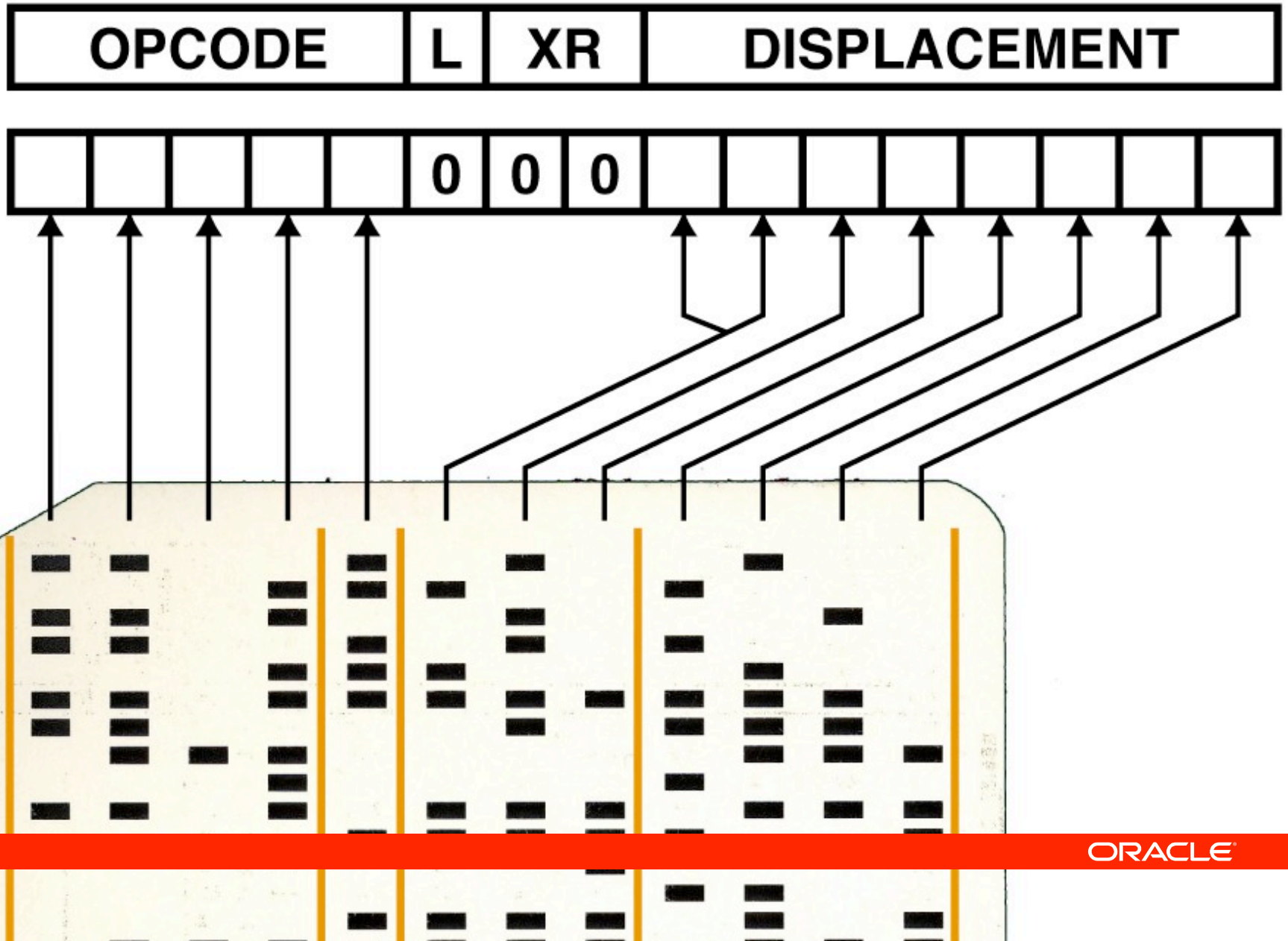
Program Load Card Input



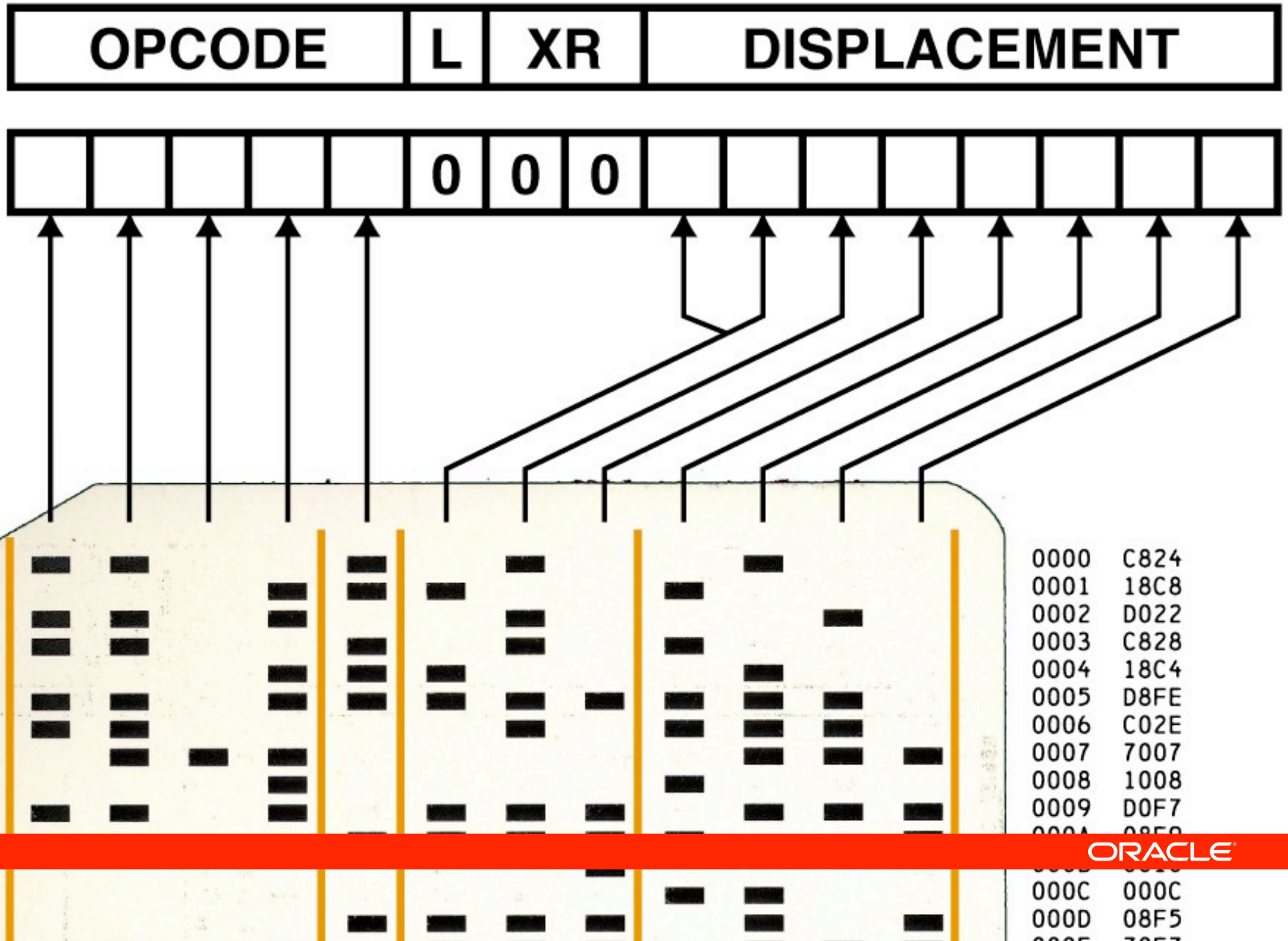
IBM 1130 Instruction Format



Draw Separator Lines



Transcribe into Hexadecimal



Analyzing the Code

```
0000  C824
0001  18C8
0002  D022
0003  C828
0004  18C4
0005  D8FE
0006  C02E
0007  7007
0008  1008
0009  D0F7
000A  08F9
000B  0010
000C  000C
000D  08F5
000E  70F7
000F  1801
0010  D0F5
0011  D01A
0012  C837
0013  18CA
0014  D835
0015  C038
0016  1804
0017  D0EB
0018  0831
0019  C0FD
001A  90E9
001B  D0FB
001C  4830
001D  7007
001E  C0EC
001F  D0F7
0020  C82C
0021  18CB
0022  40E4
0023  C0F4
0024  4008
0025  80C4
0026  0018
0027  4005
0028  C0EF
0029  80DA
002A  D0EB
002B  70ED
002C  0010
002D  9000
002E  18D0
002F  C0F2
0030  D0E5
0031  1825
0032  10C4
0033  801B
0034  D000
0035  98C0
0036  40D0
0037  C0DE
0038  90F9
0039  D0DC
003A  4830
003B  70F5
003C  C0F4
003D  40C9
003E  70ED
003F  00C4
0040  00FC
0041  00D8
0042  00DC
0043  00F0
0044  00F4
0045  00D0
0046  00D4
0047  00E4
0048  00E0
0049  003C
004A  60E8
004B  7000
004C  0030
004D  4834
004E  F010
004F  C009
```

Preliminary Disassembly

```
0000  C824  START LDD   GETWD
0001  18C8  CHAR  RTE   8
0002  D022        STO   GETWD
0003  C828  SENSE LDD   WDRET
0004  18C4  WRITE RTE   4
0005  D8FE        STD   WRITE
0006  C02E  CHRET LD    PREBR
0007  7007  CHPRT MDX   MORE
0008  1008        SLA   8
0009  D0F7        STO   CHAR
000A  08F9        XIO   WRITE
000B  0010  SIXTN DC    16
000C  000C  LEVL4 DC    LEVL4
000D  08F5        XIO   SENSE
000E  70F7        MDX   CHRET
000F  1801  MORE  SRA   1
0010  D0F5        STO   CHRET
0011  D01A        STO   WDRET
0012  C837        LDD   SWTCH
0013  18CA        RTE   10
0014  D835        STD   SWTCH
0015  C038        LD    FHACK
0016  1804  DGCNT SRA   4
0017  D0EB  WDCNT STO   SENSE
0018  0831  LOC   XIO   SWTCH
0019  C0FD  LOOP  LD    WDCNT
001A  90E9        S     WRITE
001B  D0FB        STO   WDCNT
001C  4830        BSC   -Z
001D  7007        MDX   GETWD
001E  C0EC        LD    SIXTN
001F  D0F7        STO   WDCNT
0020  C82C        LDD   EHACK
0021  18CB        RTE   11
0022  40E4  XFOUR BSI   CHPRT
0023  C0F4        LD    LOC
0024  4008        BSI   WDPRT
0025  80C4  GETWD DC    /80C4
0026  0018        DC    LOC
0027  4005        BSI   WDPRT
0028  C0EF        LD    LOC
0029  80DA        A     WRITE
002A  D0EB        STO   LOC
002B  70ED        MDX   LOOP
002C  0010  WDRET DC    /0010
002D  9000  WDPRT DC    /9000
002E  18D0        RTE   16
002F  C0F2        LD    XFOUR
0030  D0E5        STO   DGCNT
0031  1825  AGAIN SRA   37
0032  10C4  XONE  SLC   4
0033  801B        A     LHACK
0034  D000        STO   PREBR
0035  98C0  PREBR DC    /98C0
0036  40D0        BSI   CHPRT
0037  C0DE        LD    DGCNT
0038  90F9        S     XONE
0039  D0DC        STO   DGCNT
003A  4830        BSC   -Z
003B  70F5        MDX   AGAIN
003C  C0F4        LD    AGAIN
003D  40C9        BSI   CHPRT
003E  70ED        MDX   WDRET
003F  00C4  TABLE DC    /00C4      '0'
0040  00FC        DC    /00FC      '1'
0041  00D8        DC    /00D8      '2'
0042  00DC        DC    /00DC      '3'
0043  00F0        DC    /00F0      '4'
0044  00F4        DC    /00F4      '5'
0045  00D0        DC    /00D0      '6'
0046  00D4        DC    /00D4      '7'
0047  00E4        DC    /00E4      '8'
0048  00E0        DC    /00E0      '9'
0049  003C        DC    /003C      'A'
004A  60E8  SWTCH DC    /60E8      INVALID
004B  7000        DC    /7000      '.'
004C  0030        DC    /0030      'D'
004D  4834  EHACK DC    /4834      'E'
004E  F010  FHACK DC    /F010      'F'
004F  C009  LHACK LD    *-PREBR+TABLE-1
```

Initialization Code

```
0000  C824  START LDD   GETWD
0001  18C8  CHAR  RTE   8
0002  D022        STO   GETWD
0003  C828  SENSE LDD   WDRET
0004  18C4  WRITE RTE   4
0005  D8FE        STD   WRITE
0006  C02E  CHRET LD    PREBR
0007  7007  CHPRT MDX   MORE
0008  1008        SLA   8
0009  D0F7        STO   CHAR
000A  08F9        XIO   WRITE
000B  0010  SIXTN DC    16
000C  000C  LEVL4 DC    LEVL4
000D  08F5        XIO   SENSE
000E  70F7        MDX   CHRET
000F  1801  MORE  SRA   1
0010  D0F5        STO   CHRET
0011  D01A        STO   WDRET
0012  C837        LDD   SWTCH
0013  18CA        RTE   10
0014  D835        STD   SWTCH
0015  C038        LD    FHACK
0016  1804  DGCNT SRA   4
0017  D0EB  WDCNT STO   SENSE
0018  0831  LOC   XIO   SWTCH
0019  C0FD  LOOP  LD    WDCNT
001A  90E9        S     WRITE
001B  D0FB        STO   WDCNT
001C  4830        BSC   -Z
001D  7007        MDX   GETWD
001E  C0EC        LD    SIXTN
001F  D0F7        STO   WDCNT
0020  C82C        LDD   EHACK
0021  18CB        RTE   11
0022  40E4  XFOUR BSI   CHPRT
0023  C0F4        LD    LOC
0024  4008        BSI   WDPRT
0025  80C4  GETWD DC    /80C4
0026  0018        DC    LOC
0027  4005        BSI   WDPRT
0028  C0EF        LD    LOC
0029  80DA        A     WRITE
002A  D0EB        STO   LOC
002B  70ED        MDX   LOOP
002C  0010  WDRET DC    /0010
002D  9000  WDPRT DC    /9000
002E  18D0        RTE   16
002F  C0F2        LD    XFOUR
0030  D0E5        STO   DGCNT
0031  1825  AGAIN SRA   37
0032  10C4  XONE  SLC   4
0033  801B        A     LHACK
0034  D000        STO   PREBR
0035  98C0  PREBR DC    /98C0
0036  40D0        BSI   CHPRT
0037  C0DE        LD    DGCNT
0038  90F9        S     XONE
0039  D0DC        STO   DGCNT
003A  4830        BSC   -Z
003B  70F5        MDX   AGAIN
003C  C0F4        LD    AGAIN
003D  40C9        BSI   CHPRT
003E  70ED        MDX   WDRET
003F  00C4  TABLE DC    /00C4      '0'
0040  00FC        DC    /00FC      '1'
0041  00D8        DC    /00D8      '2'
0042  00DC        DC    /00DC      '3'
0043  00F0        DC    /00F0      '4'
0044  00F4        DC    /00F4      '5'
0045  00D0        DC    /00D0      '6'
0046  00D4        DC    /00D4      '7'
0047  00E4        DC    /00E4      '8'
0048  00E0        DC    /00E0      '9'
0049  003C        DC    /003C      'A'
004A  60E8  SWTCH DC    /60E8      INVALID
004B  7000        DC    /7000      '.'
004C  0030        DC    /0030      'D'
004D  4834  EHACK DC    /4834      'E'
004E  F010  FHACK DC    /F010      'F'
004F  C009  LHACK LD    *-PREBR+TABLE-1
```

Rewritten Code

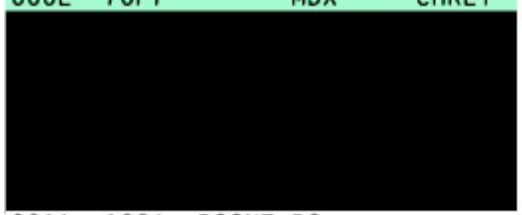
```
0000  C824  START LDD     GETWD
0001  18C8  CHAR  DC      *-*
0002  D022        STO     GETWD
0003  0F01  SENSE DC      /0F01
0004  0001  WRITE DC      CHAR
0005  0900        DC      /0900
0006  4C60  CHRET BOSC L  Z
0007  7007  CHPRT DC      *-*
0008  1008        SLA     8
0009  D0F7        STO     CHAR
000A  08F9        XIO     WRITE
000B  0010  SIXTN DC      16
000C  000C  LEVL4 DC      LEVL4
000D  08F5        XIO     SENSE
000E  70F7        MDX     CHRET
000F  1801  MORE  SRA     1
0010  D0F5        STO     CHRET
0011  D01A        STO     WDRET
0012  C837        LDD     SWTCH
0013  18CA        RTE     10
0014  D835        STD     SWTCH
0015  C038        LD      FHACK
0016  1804  DGCNT DC      *-*
0017  D0EB  WDCNT STO     SENSE
0018  NNNN  LOC   DC      /NNNN
0019  C0FD  LOOP  LD      WDCNT
001A  90E9        S       WRITE
001B  D0FB        STO     WDCNT
001C  4830        BSC     -Z
001D  7007        MDX     GETWD
001E  C0EC        LD      SIXTN
001F  D0F7        STO     WDCNT
0020  C82C        LDD     EHACK
0021  18CB        RTE     11
0022  40E4  XFOUR BSI     CHPRT
0023  C0F4        LD      LOC
0024  4008        BSI     WDPRT
0025  C480  GETWD LD    I
0026  0018              LOC
0027  4005        BSI     WDPRT
0028  C0EF        LD      LOC
0029  80DA        A       WRITE
002A  D0EB        STO     LOC
002B  70ED        MDX     LOOP
002C  4C60  WDRET BOSC L  Z
002D  9000  WDPRT DC      *-*
002E  18D0        RTE     16
002F  C0F2        LD      XFOUR
0030  D0E5        STO     DGCNT
0031  1825  AGAIN SRA     37
0032  10C4  XONE  SLC     4
0033  801B        A       LHACK
0034  D000        STO     FETCH
0035  98C0  FETCH DC      *-*
0036  40D0        BSI     CHPRT
0037  C0DE        LD      DGCNT
0038  90F9        S       XONE
0039  D0DC        STO     DGCNT
003A  4830        BSC     -Z
003B  70F5        MDX     AGAIN
003C  C0F4        LD      AGAIN
003D  40C9        BSI     CHPRT
003E  70ED        MDX     WDRET
003F  00C4  TABLE DC      /00C4     '0'
0040  00FC        DC      /00FC     '1'
0041  00D8        DC      /00D8     '2'
0042  00DC        DC      /00DC     '3'
0043  00F0        DC      /00F0     '4'
0044  00F4        DC      /00F4     '5'
0045  00D0        DC      /00D0     '6'
0046  00D4        DC      /00D4     '7'
0047  00E4        DC      /00E4     '8'
0048  00E0        DC      /00E0     '9'
0049  003C        DC      /003C     'A'
004A  0018  SWTCH DC      /0018     'B'
004B  3A1C        DC      /3A1C     'C'
004C  0030        DC      /0030     'D'
004D  4834  EHACK DC      /4834     'E'
004E  F010  FHACK DC      /F010     'F'
004F  C009  LHACK LD      *-PREBR+TABLE-1
```

Code No Longer Needed

```
0001  18C8  CHAR   DC       *-*
0003  0F01  SENSE  DC       /0F01
0004  0001  WRITE  DC       CHAR
0005  0900         DC       /0900
0006  4C60  CHRET  BOSC L   Z
0007  7007  CHPRT  DC       *-*
0008  1008         SLA      8
0009  D0F7         STO      CHAR
000A  08F9         XIO      WRITE
000B  0010  SIXTN  DC       16
000C  000C  LEVL4  DC       LEVL4
000D  08F5         XIO      SENSE
000E  70F7         MDX      CHRET
0016  1804  DGCNT  DC       *-*
0017  D0EB  WDCNT  STO      SENSE
0018  NNNN  LOC    DC       /NNNN
0019  C0FD  LOOP   LD       WDCNT
001A  90E9         S        WRITE
001B  D0FB         STO      WDCNT
001C  4830         BSC      -Z
001D  7007         MDX      GETWD
001E  C0EC         LD       SIXTN
001F  D0F7         STO      WDCNT
0020  C82C         LDD      EHACK
0021  18CB         RTE      11
0022  40E4  XFOUR  BSI      CHPRT
0023  C0F4         LD       LOC
0024  4008         BSI      WDPRT
0025  C480  GETWD  LD    I
0026  0018                  LOC
0027  4005         BSI      WDPRT
0028  C0EF         LD       LOC
0029  80DA         A        WRITE
002A  D0EB         STO      LOC
002B  70ED         MDX      LOOP
002C  4C60  WDRET  BOSC L   Z
002D  9000  WDPRT  DC       *-*
002E  18D0         RTE      16
002F  C0F2         LD       XFOUR
0030  D0E5         STO      DGCNT
0031  1825  AGAIN  SRA      37
0032  10C4  XONE   SLC      4
0033  801B         A        LHACK
0034  D000         STO      FETCH
0035  98C0  FETCH  DC       *-*
0036  40D0         BSI      CHPRT
0037  C0DE         LD       DGCNT
0038  90F9         S        XONE
0039  D0DC         STO      DGCNT
003A  4830         BSC      -Z
003B  70F5         MDX      AGAIN
003C  C0F4         LD       AGAIN
003D  40C9         BSI      CHPRT
003E  70ED         MDX      WDRET
003F  00C4  TABLE  DC       /00C4      '0'
0040  00FC         DC       /00FC      '1'
0041  00D8         DC       /00D8      '2'
0042  00DC         DC       /00DC      '3'
0043  00F0         DC       /00F0      '4'
0044  00F4         DC       /00F4      '5'
0045  00D0         DC       /00D0      '6'
0046  00D4         DC       /00D4      '7'
0047  00E4         DC       /00E4      '8'
0048  00E0         DC       /00E0      '9'
0049  003C         DC       /003C      'A'
004A  0018  SWTCH  DC       /0018      'B'
004B  3A1C         DC       /3A1C      'C'
004C  0030         DC       /0030      'D'
004D  4834  EHACK  DC       /4834      'E'
004E  F010  FHACK  DC       /F010      'F'
004F  C009  LHACK  LD       *-PREBR+TABLE-1
```

Main Program, Two Subroutines

```
0001  18C8  CHAR   DC       *-*

0003  0F01  SENSE  DC       /0F01
0004  0001  WRITE  DC       CHAR
0005  0900         DC       /0900
0006  4C60  CHRET  BOSC L   Z
0007  7007  CHPRT  DC       *-*
0008  1008         SLA      8
0009  D0F7         STO      CHAR
000A  08F9         XIO      WRITE
000B  0010  SIXTN  DC       16
000C  000C  LEVL4  DC       LEVL4
000D  08F5         XIO      SENSE
000E  70F7         MDX      CHRET
```



```
0016  1804  DGCNT  DC       *-*
0017  D0EB  WDCNT  STO      SENSE
0018  NNNN  LOC    DC       /NNNN
0019  C0FD  LOOP   LD       WDCNT
001A  90E9         S        WRITE
001B  D0FB         STO      WDCNT
001C  4830         BSC      -Z
001D  7007         MDX      GETWD
001E  C0EC         LD       SIXTN
001F  D0F7         STO      WDCNT
0020  C82C         LDD      EHACK
0021  18CB         RTE      11
0022  40E4  XFOUR  BSI      CHPRT
0023  C0F4         LD       LOC
0024  4008         BSI      WDPRT
0025  C480  GETWD  LD    I
0026  0018                  LOC
0027  4005         BSI      WDPRT
0028  C0EF         LD       LOC
0029  80DA         A        WRITE
002A  D0EB         STO      LOC
002B  70ED         MDX      LOOP
002C  4C60  WDRET  BOSC L   Z
002D  9000  WDPRT  DC       *-*
002E  18D0         RTE      16
002F  C0F2         LD       XFOUR
0030  D0E5         STO      DGCNT
0031  1825  AGAIN  SRA      37
0032  10C4  XONE   SLC      4
0033  801B         A        LHACK
0034  D000         STO      FETCH
0035  98C0  FETCH  DC       *-*
0036  40D0         BSI      CHPRT
0037  C0DE         LD       DGCNT
0038  90F9         S        XONE
0039  D0DC         STO      DGCNT
003A  4830         BSC      -Z
003B  70F5         MDX      AGAIN
003C  C0F4         LD       AGAIN
003D  40C9         BSI      CHPRT
003E  70ED         MDX      WDRET
003F  00C4  TABLE  DC       /00C4      '0'
0040  00FC         DC       /00FC      '1'
0041  00D8         DC       /00D8      '2'
0042  00DC         DC       /00DC      '3'
0043  00F0         DC       /00F0      '4'
0044  00F4         DC       /00F4      '5'
0045  00D0         DC       /00D0      '6'
0046  00D4         DC       /00D4      '7'
0047  00E4         DC       /00E4      '8'
0048  00E0         DC       /00E0      '9'
0049  003C         DC       /003C      'A'
004A  0018  SWTCH  DC       /0018      'B'
004B  3A1C         DC       /3A1C      'C'
004C  0030         DC       /0030      'D'
004D  4834  EHACK  DC       /4834      'E'
004E  F010  FHACK  DC       /F010      'F'
004F  C009  LHACK  LD       *-PREBR+TABLE-1
```

Subroutine: Print Character

```
0001   18C8   CHAR   DC          *-*

0003   0F01   SENSE  DC          /0F01
0004   0001   WRITE  DC          CHAR
0005   0900          DC          /0900
0006   4C60   CHRET  BOSC  L     Z
0007   7007   CHPRT  DC          *-*
0008   1008          SLA         8
0009   D0F7          STO         CHAR
000A   08F9          XIO         WRITE
000B   0010   SIXTN  DC          16
000C   000C   LEVL4  DC          LEVL4
000D   08F5          XIO         SENSE
000E   70F7          MDX         CHRET
```

Write Character to Printer

```
0001   18C8   CHAR   DC        *-*

0003   0F01   SENSE  DC        /0F01
0004   0001   WRITE  DC        CHAR
0005   0900          DC        /0900
0006   4C60   CHRET  BOSC  L   Z
0007   7007   CHPRT  DC        *-*
0008   1008          SLA       8
0009   D0F7          STO       CHAR
000A   08F9          XIO       WRITE
000B   0010   SIXTN  DC        16
000C   000C   LEVL4  DC        LEVL4
000D   08F5          XIO       SENSE
000E   70F7          MDX       CHRET
```

Wait for Interrupt

```
0001   18C8   CHAR   DC        *-*

0003   0F01   SENSE  DC        /0F01
0004   0001   WRITE  DC        CHAR
0005   0900          DC        /0900
0006   4C60   CHRET  BOSC  L   Z
0007   7007   CHPRT  DC        *-*
0008   1008          SLA       8
0009   D0F7          STO       CHAR
000A   08F9          XIO       WRITE
000B   0010   SIXTN  DC        16
000C   000C   LEVL4  DC        LEVL4
000D   08F5          XIO       SENSE
000E   70F7          MDX       CHRET
```

IBM 1130 Interrupt Vector

```
0001   18C8   CHAR   DC          *-*

0003   0F01   SENSE  DC          /0F01
0004   0001   WRITE  DC          CHAR
0005   0900          DC          /0900
0006   4C60   CHRET  BOSC  L     Z
0007   7007   CHPRT  DC          *-*
0008   1008          SLA         8
0009   D0F7          STO         CHAR
000A   08F9          XIO         WRITE
000B   0010   SIXTN  DC          16
000C   000C   LEVL4  DC          LEVL4
000D   08F5          XIO         SENSE
000E   70F7          MDX         CHRET
```

Level 4 Interrupt Address

```
0001   18C8   CHAR   DC          *-*

0003   0F01   SENSE  DC          /0F01
0004   0001   WRITE  DC          CHAR
0005   0900          DC          /0900
0006   4C60   CHRET  BOSC  L     Z
0007   7007   CHPRT  DC          *-*
0008   1008          SLA         8
0009   D0F7          STO         CHAR
000A   08F9          XIO         WRITE
000B   0010   SIXTN  DC          16
000C   000C   LEVL4  DC          LEVL4
000D   08F5          XIO         SENSE
000E   70F7          MDX         CHRET
```

Return (and Dismiss Interrupt)

```
0001   18C8   CHAR   DC        *-*
0003   0F01   SENSE  DC        /0F01
0004   0001   WRITE  DC        CHAR
0005   0900          DC        /0900
0006   4C60   CHRET  BOSC  L    Z
0007   7007   CHPRT  DC        *-*
0008   1008          SLA       8
0009   D0F7          STO       CHAR
000A   08F9          XIO       WRITE
000B   0010   SIXTN  DC        16
000C   000C   LEVL4  DC        LEVL4
000D   08F5          XIO       SENSE
000E   70F7          MDX       CHRET
```

Main Program: Counters

```
0001  18C8  CHAR   DC        *-*

0003  0F01  SENSE  DC        /0F01
0004  0001  WRITE  DC        CHAR
0005  0900         DC        /0900
0006  4C60  CHRET  BOSC L    Z
0007  7007  CHPRT  DC        *-*
0008  1008         SLA       8
0009  D0F7         STO       CHAR
000A  08F9         XIO       WRITE
000B  0010  SIXTN  DC        16
000C  000C  LEVL4  DC        LEVL4
000D  08F5         XIO       SENSE
000E  70F7         MDX       CHRET

0016  1804  DGCNT  DC        *-*
0017  D0EB  WDCNT  STO       SENSE
0018  NNNN  LOC    DC        /NNNN
```

```
0019  C0FD  LOOP   LD        WDCNT
001A  90E9         S         WRITE
001B  D0FB         STO       WDCNT
001C  4830         BSC       -Z
001D  7007         MDX       GETWD
001E  C0EC         LD        SIXTN
001F  D0F7         STO       WDCNT
0020  C82C         LDD       EHACK
0021  18CB         RTE       11
0022  40E4  XFOUR  BSI       CHPRT
0023  C0F4         LD        LOC
0024  4008         BSI       WDPRT
0025  C480  GETWD  LD    I
0026  0018                   LOC
0027  4005         BSI       WDPRT
0028  C0EF         LD        LOC
0029  80DA         A         WRITE
002A  D0EB         STO       LOC
002B  70ED         MDX       LOOP

004D  4834  EHACK  DC        /4834        'E'
```

How the Counters Work

```
0001   18C8   CHAR   DC       *-*

0003   0F01   SENSE  DC       /0F01
0004   0001   WRITE  DC       CHAR
0005   0900          DC       /0900
0006   4C60   CHRET  BOSC L   Z
0007   7007   CHPRT  DC       *-*
0008   1008          SLA      8
0009   D0F7          STO      CHAR
000A   08F9          XIO      WRITE
000B   0010   SIXTN  DC       16
000C   000C   LEVL4  DC       LEVL4
000D   08F5          XIO      SENSE
000E   70F7          MDX      CHRET

0016   1804   DGCNT  DC       *-*
0017   D0EB   WDCNT  STO      SENSE
0018   NNNN   LOC    DC       /NNNN
```

```
0019   C0FD   LOOP   LD       WDCNT
001A   90E9          S        WRITE
001B   D0FB          STO      WDCNT
001C   4830          BSC      -Z
001D   7007          MDX      GETWD
001E   C0EC          LD       SIXTN
001F   D0F7          STO      WDCNT
0020   C82C          LDD      EHACK
0021   18CB          RTE      11
0022   40E4   XFOUR  BSI      CHPRT
0023   C0F4          LD       LOC
0024   4008          BSI      WDPRT
0025   C480   GETWD  LD    I
0026   0018                   LOC
0027   4005          BSI      WDPRT
0028   C0EF          LD       LOC
0029   80DA          A        WRITE
002A   D0EB          STO      LOC
002B   70ED          MDX      LOOP

004D   4834   EHACK  DC       /4834        'E'
```

Print Newline and Shift to Red

```
0001  18C8  CHAR   DC       *-*

0003  0F01  SENSE  DC       /0F01
0004  0001  WRITE  DC       CHAR
0005  0900         DC       /0900
0006  4C60  CHRET  BOSC L   Z
0007  7007  CHPRT  DC       *-*
0008  1008         SLA      8
0009  D0F7         STO      CHAR
000A  08F9         XIO      WRITE
000B  0010  SIXTN  DC       16
000C  000C  LEVL4  DC       LEVL4
000D  08F5         XIO      SENSE
000E  70F7         MDX      CHRET

0016  1804  DGCNT  DC       *-*
0017  D0EB  WDCNT  STO      SENSE
0018  NNNN  LOC    DC       /NNNN
```

```
0019  C0FD  LOOP   LD       WDCNT
001A  90E9         S        WRITE
001B  D0FB         STO      WDCNT
001C  4830         BSC      -Z
001D  7007         MDX      GETWD
001E  C0EC         LD       SIXTN
001F  D0F7         STO      WDCNT
0020  C82C         LDD      EHACK
0021  18CB         RTE      11
0022  40E4  XFOUR  BSI      CHPRT
0023  C0F4         LD       LOC
0024  4008         BSI      WDPRT
0025  C480  GETWD  LD    I
0026  0018                  LOC
0027  4005         BSI      WDPRT
0028  C0EF         LD       LOC
0029  80DA         A        WRITE
002A  D0EB         STO      LOC
002B  70ED         MDX      LOOP

004D  4834  EHACK  DC       /4834      'E'
```

Subroutine: Print Word

```
001C  4830        BSC     -Z
001D  7007        MDX     GETWD
001E  C0EC        LD      SIXTN
001F  D0F7        STO     WDCNT
0020  C82C        LDD     EHACK
0021  18CB        RTE     11
0022  40E4  XFOUR BSI     CHPRT
0023  C0F4        LD      LOC
0024  4008        BSI     WDPRT
0025  C480  GETWD LD    I
0026  0018                LOC
0027  4005        BSI     WDPRT
0028  C0EF        LD      LOC
0029  80DA        A       WRITE
002A  D0EB        STO     LOC
002B  70ED        MDX     LOOP
```

```
002C  4C60  WDRET BOSC L  Z
002D  9000  WDPRT DC      *-*
002E  18D0        RTE     16
002F  C0F2        LD      XFOUR
0030  D0E5        STO     DGCNT
0031  1825  AGAIN SRA     37
0032  10C4  XONE  SLC     4
0033  801B        A       LHACK
0034  D000        STO     FETCH
0035  98C0  FETCH DC      *-*
0036  40D0        BSI     CHPRT
0037  C0DE        LD      DGCNT
0038  90F9        S       XONE
0039  D0DC        STO     DGCNT
003A  4830        BSC     -Z
003B  70F5        MDX     AGAIN
003C  C0F4        LD      AGAIN
003D  40C9        BSI     CHPRT
003E  70ED        MDX     WDRET
```

Counting Four Hex Digits

```
001C  4830         BSC      -Z
001D  7007         MDX      GETWD
001E  C0EC         LD       SIXTN
001F  D0F7         STO      WDCNT
0020  C82C         LDD      EHACK
0021  18CB         RTE      11
0022  40E4  XFOUR  BSI      CHPRT
0023  C0F4         LD       LOC
0024  4008         BSI      WDPRT
0025  C480  GETWD  LD    I
0026  0018                  LOC
0027  4005         BSI      WDPRT
0028  C0EF         LD       LOC
0029  80DA         A        WRITE
002A  D0EB         STO      LOC
002B  70ED         MDX      LOOP
```

```
002C  4C60  WDRET  BOSC L   Z
002D  9000  WDPRT  DC       *-*
002E  18D0         RTE      16
002F  C0F2         LD       XFOUR
0030  D0E5         STO      DGCNT
0031  1825  AGAIN  SRA      37
0032  10C4  XONE   SLC      4
0033  801B         A        LHACK
0034  D000         STO      FETCH
0035  98C0  FETCH  DC       *-*
0036  40D0         BSI      CHPRT
0037  C0DE         LD       DGCNT
0038  90F9         S        XONE
0039  D0DC         STO      DGCNT
003A  4830         BSC      -Z
003B  70F5         MDX      AGAIN
003C  C0F4         LD       AGAIN
003D  40C9         BSI      CHPRT
003E  70ED         MDX      WDRET
```

How the Counter Works

```
001C  4830        BSC     -Z
001D  7007        MDX     GETWD
001E  C0EC        LD      SIXTN
001F  D0F7        STO     WDCNT
0020  C82C        LDD     EHACK
0021  18CB        RTE     11
0022  40E4  XFOUR BSI     CHPRT
0023  C0F4        LD      LOC
0024  4008        BSI     WDPRT
0025  C480  GETWD LD    I
0026  0018                LOC
0027  4005        BSI     WDPRT
0028  C0EF        LD      LOC
0029  80DA        A       WRITE
002A  D0EB        STO     LOC
002B  70ED        MDX     LOOP
```

```
002C  4C60  WDRET BOSC L  Z
002D  9000  WDPRT DC      *-*
002E  18D0        RTE     16
002F  C0F2        LD      XFOUR
0030  D0E5        STO     DGCNT
0031  1825  AGAIN SRA     37
0032  10C4  XONE  SLC     4
0033  801B        A       LHACK
0034  D000        STO     FETCH
0035  98C0  FETCH DC      *-*
0036  40D0        BSI     CHPRT
0037  C0DE        LD      DGCNT
0038  90F9        S       XONE
0039  D0DC        STO     DGCNT
003A  4830        BSC     -Z
003B  70F5        MDX     AGAIN
003C  C0F4        LD      AGAIN
003D  40C9        BSI     CHPRT
003E  70ED        MDX     WDRET
```

Clearing the Accumulator: Strange

```
001C  4830         BSC      -Z
001D  7007         MDX      GETWD
001E  C0EC         LD       SIXTN
001F  D0F7         STO      WDCNT
0020  C82C         LDD      EHACK
0021  18CB         RTE      11
0022  40E4  XFOUR  BSI      CHPRT
0023  C0F4         LD       LOC
0024  4008         BSI      WDPRT
0025  C480  GETWD  LD    I
0026  0018                  LOC
0027  4005         BSI      WDPRT
0028  C0EF         LD       LOC
0029  80DA         A        WRITE
002A  D0EB         STO      LOC
002B  70ED         MDX      LOOP
```

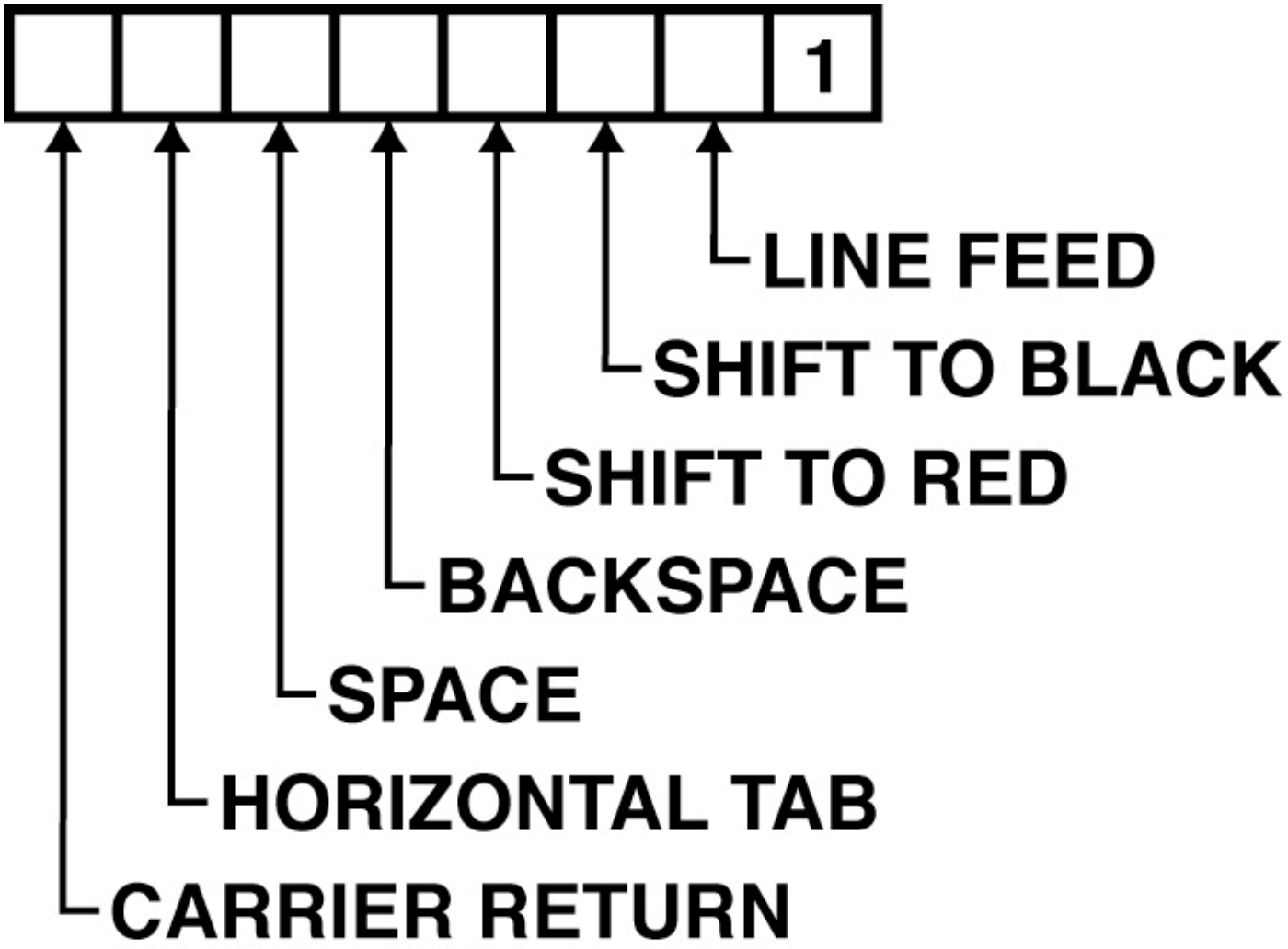
```
002C  4C60  WDRET  BOSC L   Z
002D  9000  WDPRT  DC       *-*
002E  18D0         RTE      16
002F  C0F2         LD       XFOUR
0030  D0E5         STO      DGCNT
0031  1825  AGAIN  SRA      37
0032  10C4  XONE   SLC      4
0033  801B         A        LHACK
0034  D000         STO      FETCH
0035  98C0  FETCH  DC       *-*
0036  40D0         BSI      CHPRT
0037  C0DE         LD       DGCNT
0038  90F9         S        XONE
0039  D0DC         STO      DGCNT
003A  4830         BSC      -Z
003B  70F5         MDX      AGAIN
003C  C0F4         LD       AGAIN
003D  40C9         BSI      CHPRT
003E  70ED         MDX      WDRET
```

Print Space and Shift to Black

```
001C  4830        BSC     -Z
001D  7007        MDX     GETWD
001E  C0EC        LD      SIXTN
001F  D0F7        STO     WDCNT
0020  C82C        LDD     EHACK
0021  18CB        RTE     11
0022  40E4  XFOUR BSI     CHPRT
0023  C0F4        LD      LOC
0024  4008        BSI     WDPRT
0025  C480  GETWD LD    I
0026  0018                LOC
0027  4005        BSI     WDPRT
0028  C0EF        LD      LOC
0029  80DA        A       WRITE
002A  D0EB        STO     LOC
002B  70ED        MDX     LOOP
```

```
002C  4C60  WDRET BOSC L  Z
002D  9000  WDPRT DC      *-*
002E  18D0        RTE     16
002F  C0F2        LD      XFOUR
0030  D0E5        STO     DGCNT
0031  1825  AGAIN SRA     37
0032  10C4  XONE  SLC     4
0033  801B        A       LHACK
0034  D000        STO     FETCH
0035  98C0  FETCH DC      *-*
0036  40D0        BSI     CHPRT
0037  C0DE        LD      DGCNT
0038  90F9        S       XONE
0039  D0DC        STO     DGCNT
003A  4830        BSC     -Z
003B  70F5        MDX     AGAIN
003C  C0F4        LD      AGAIN
003D  40C9        BSI     CHPRT
003E  70ED        MDX     WDRET
```

Console Printer Control Codes



The DEC PDP-10

- 36-bit words
- 16 registers
- One mega-OPS
- One megabyte
- One megabuck

Triple-angle Formula for Sine

$$\sin 3x = 3 \sin x - 4 \sin^3 x$$

Reorganized Formula for Sine

$$\sin 3x = 3 \sin x - 4 \sin^3 x$$

$$\sin a = 4 \sin^3 a/(-3) - 3 \sin a/(-3)$$

Computing Sine (HAKMEM #158)

$$\sin 3x = 3 \sin x - 4 \sin^3 x$$

$$\sin a = 4 \sin^3 a/(-3) - 3 \sin a/(-3)$$

```
SIN:  MOVM  B,A              ;argument in A
      CAMG  B,[.00017]
       POPJ P,               ;sin a = a, within 27 bits
      FVDRI A,(-3.0)
      PUSHJ P,SIN            ;sin a/(-3)
      FMPR  B,B
      FSC   B,2
      FADRI B,(-3.0)
      FMPRB A,B
      POPJ  P,               ;sin in A, sin or |sin| in B
```

How to Save One Word (9%)

$$\sin 3x = 3 \sin x - 4 \sin^3 x$$

$$\sin a = 4 \sin^3 a/(-3) - 3 \sin a/(-3)$$

```
SIN:  MOVM  B,A              ;argument in A
      CAMG  B,FOO
       POPJ P,               ;sin a = a, within 27 bits
      FVDRI A,(-3.0)
      PUSHJ P,SIN            ;sin a/(–3)
FOO:  FMPR  B,B              ;.0001678467 if B=11
      FSC   B,2
      FADRI B,(-3.0)
      FMPRB A,B
      POPJ  P,               ;sin in A, sin or |sin| in B
```

Automating Resource Management

- Coding in octal or decimal
- Assemblers
- Relocating assemblers and linkers
- Expression compilation
- Register allocation
- Stack management of local data
- Heap management
- Virtual memory / address remapping

Main Points of This Talk

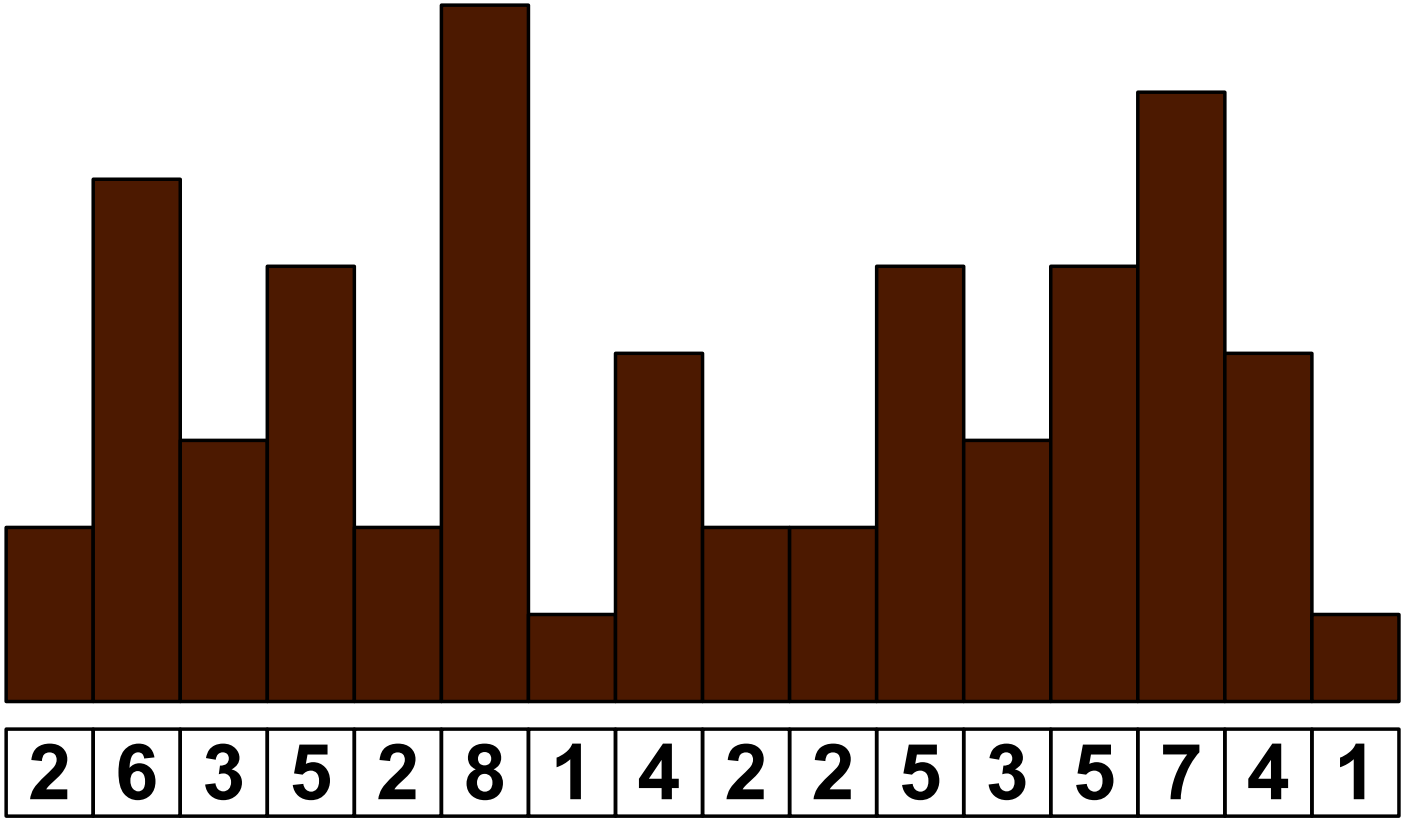
- The best way to write parallel applications is not to have to think about parallelism.
  - Need for separation of concerns
- The issue is not so much parallelism as *independence*.
- Accumulators are BAD. Divide-and-conquer is GOOD.
  - An old message, but now we need to take it seriously.
- Certain algebraic properties are very important.
  - Programmers need help to ensure these properties.
- For debugging, reproduceability is extremely important.
  - Worth sacrificing performance for (another old message)

Toy Programming Problem (1 of 3)

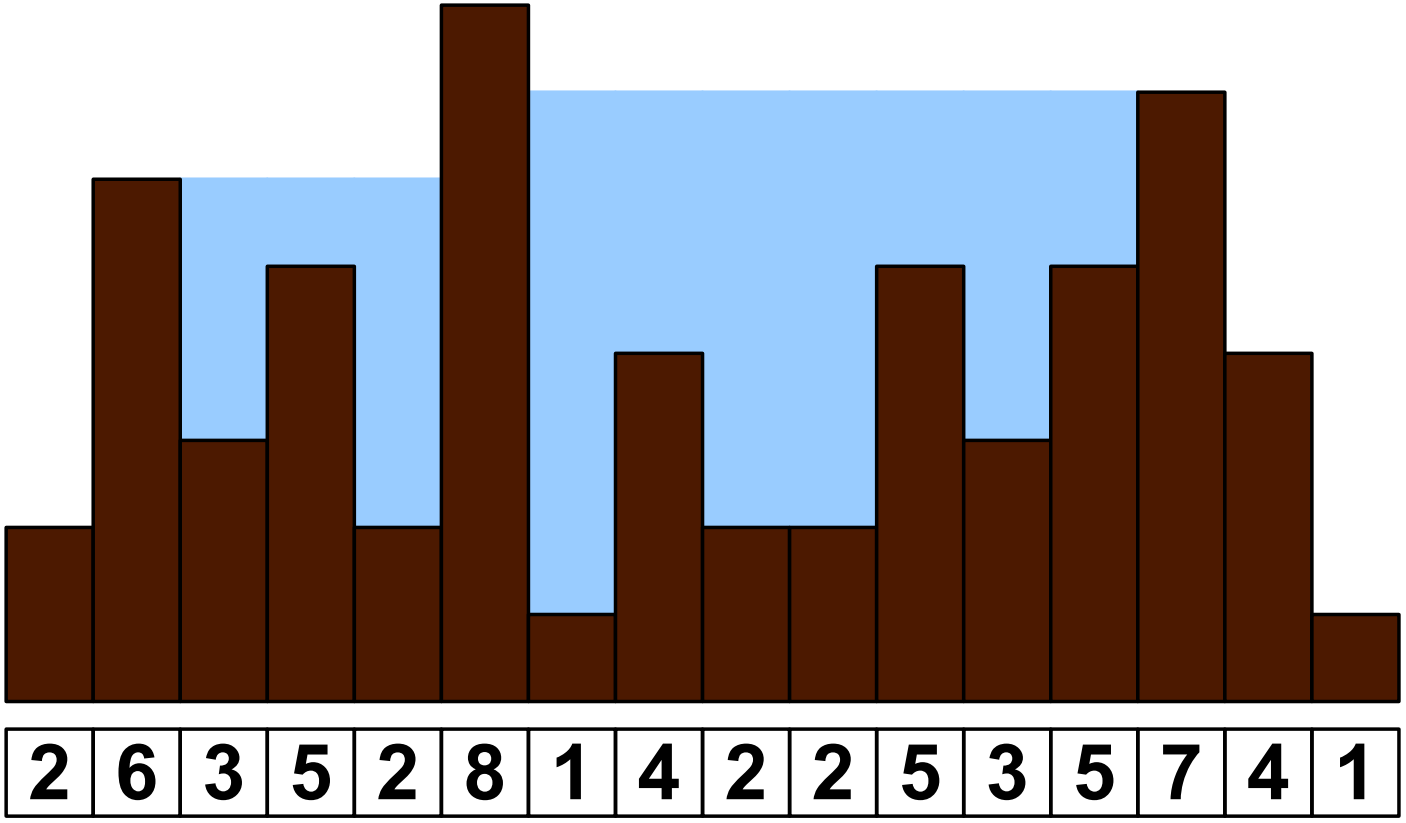
2	6	3	5	2	8	1	4	2	2	5	3	5	7	4	1

Thanks to Dan Nussbaum and Steve Heller for suggesting this toy problem.

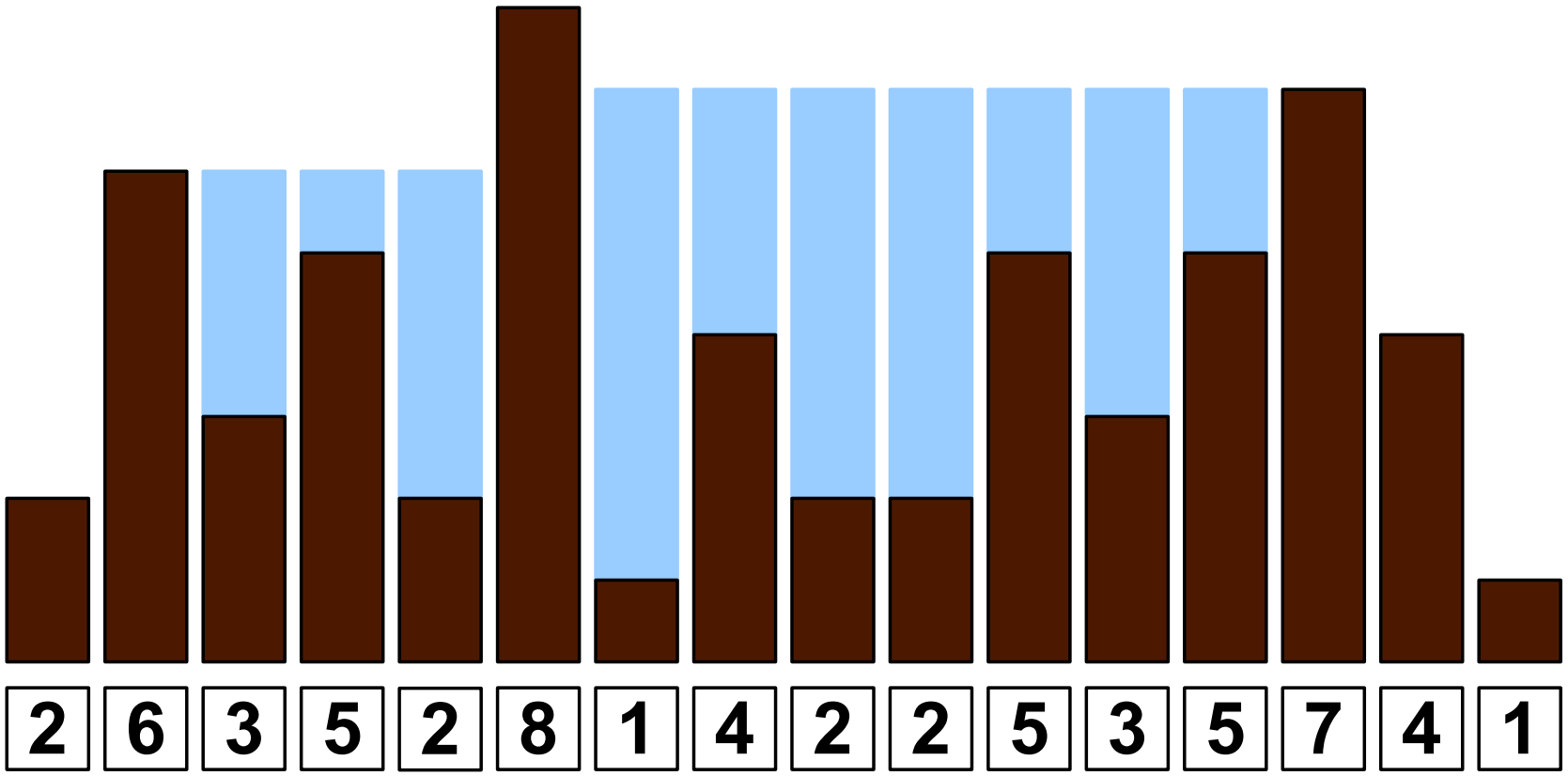
Toy Programming Problem (2 of 3)



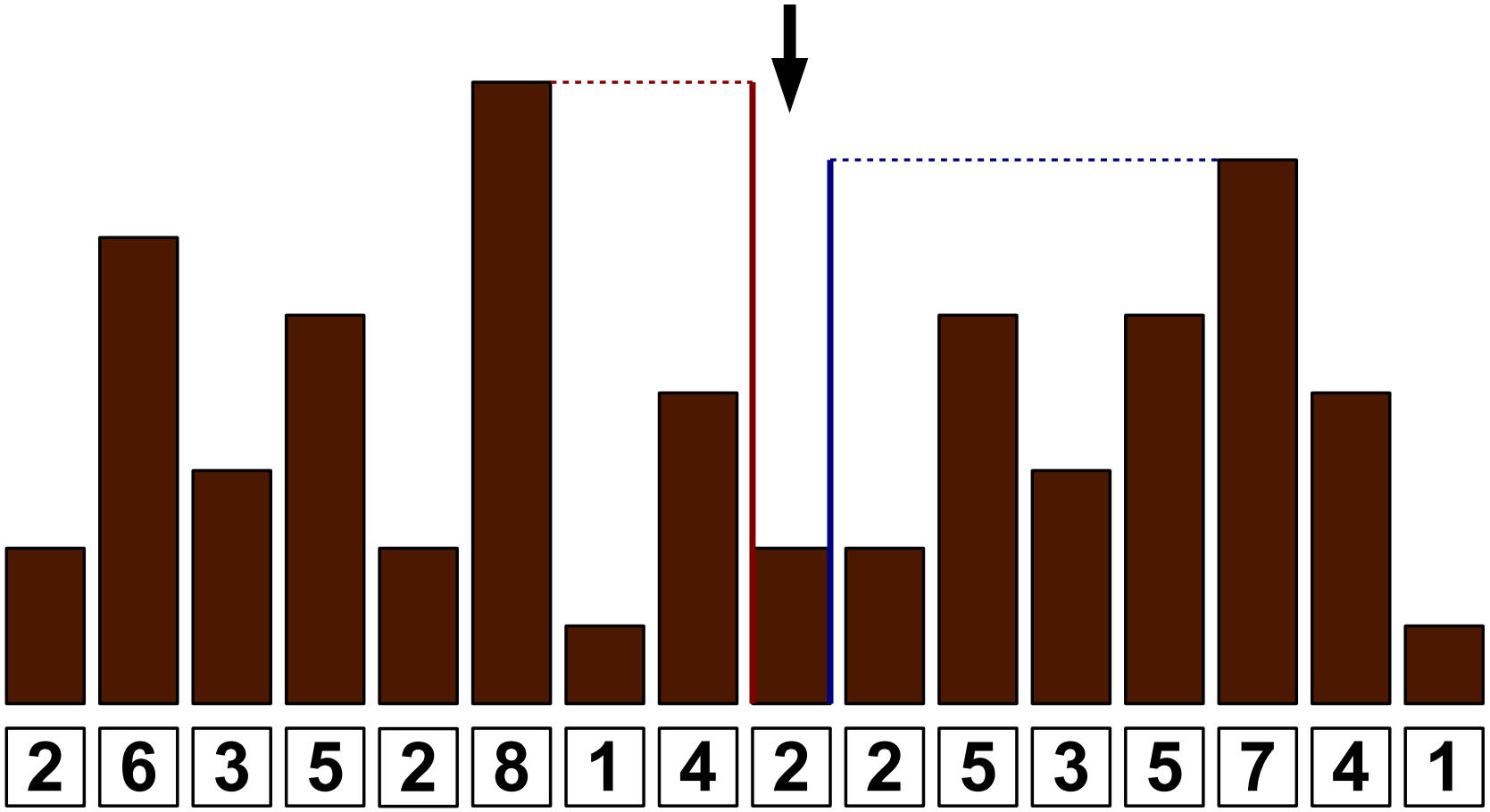
Toy Programming Problem (3 of 3)



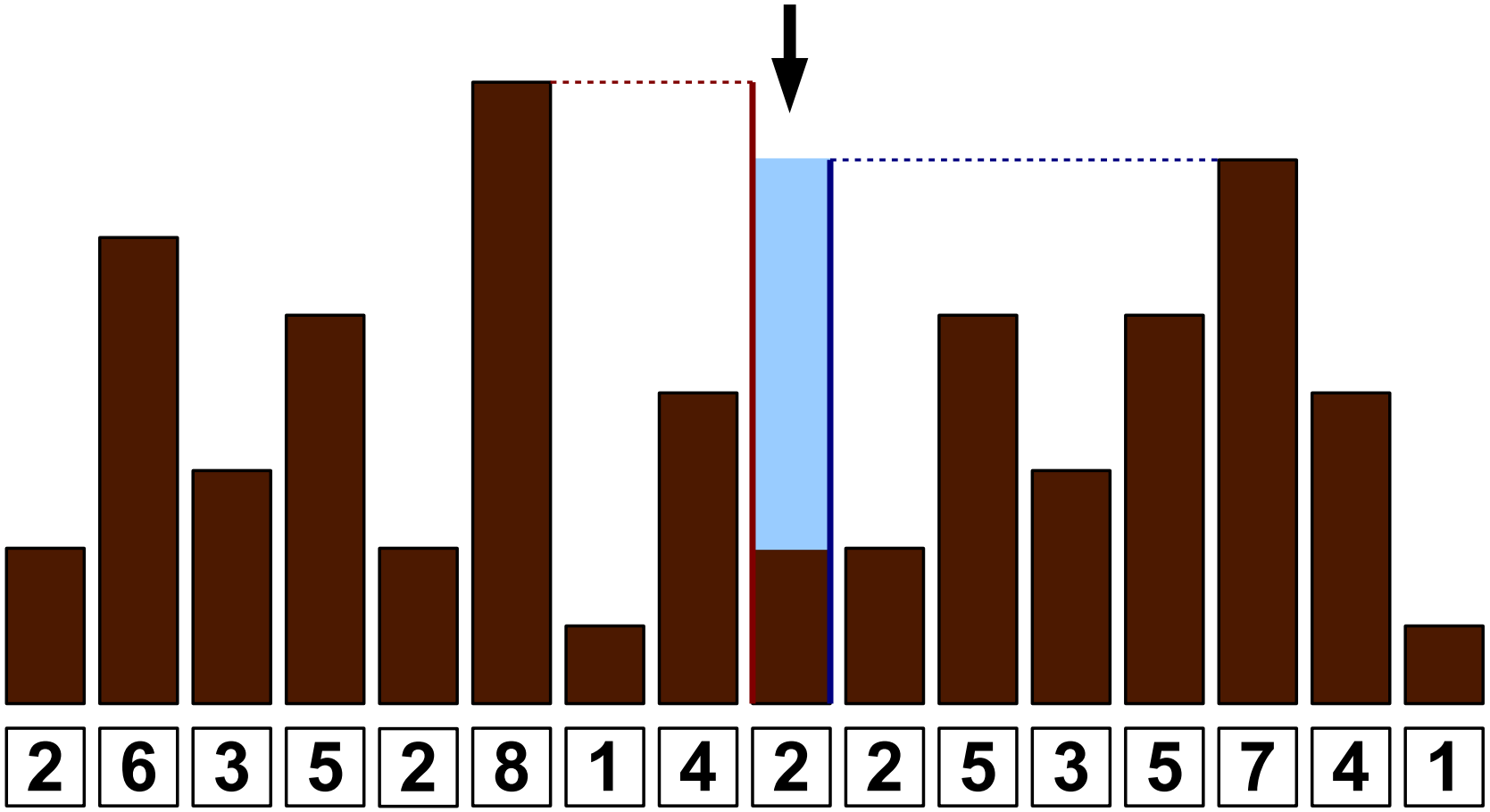
Key Insight (1 of 3)



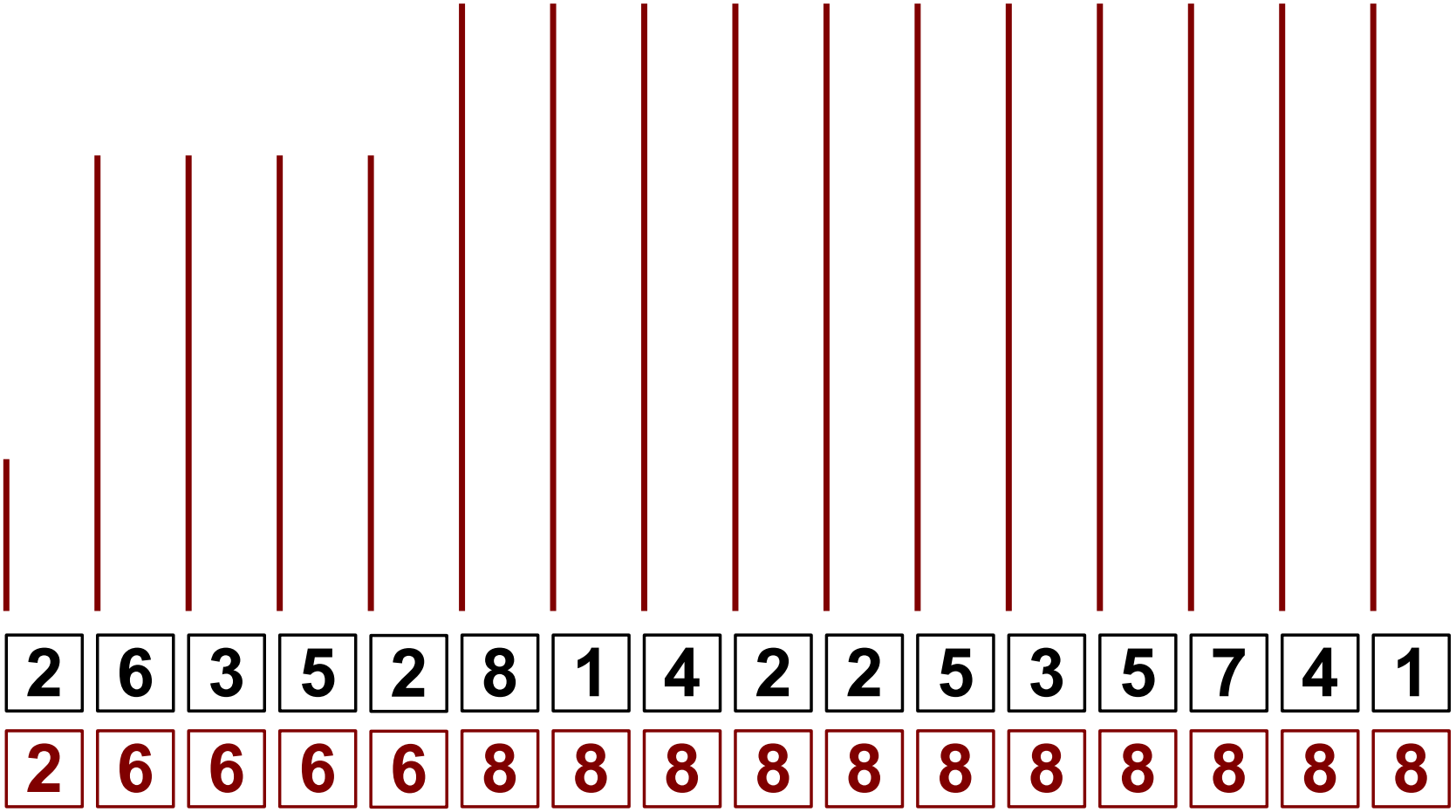
Key Insight (2 of 3)



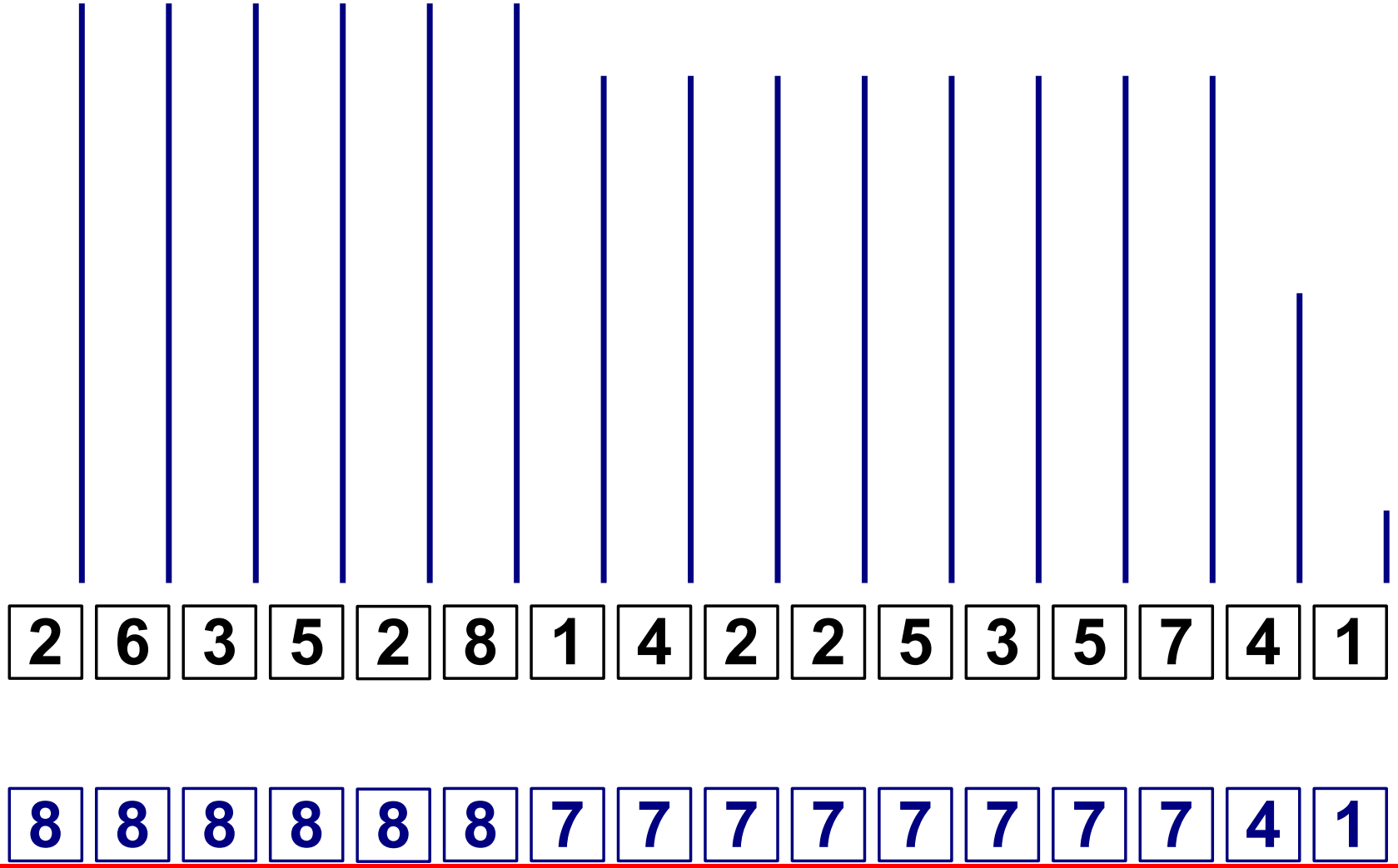
Key Insight (3 of 3)



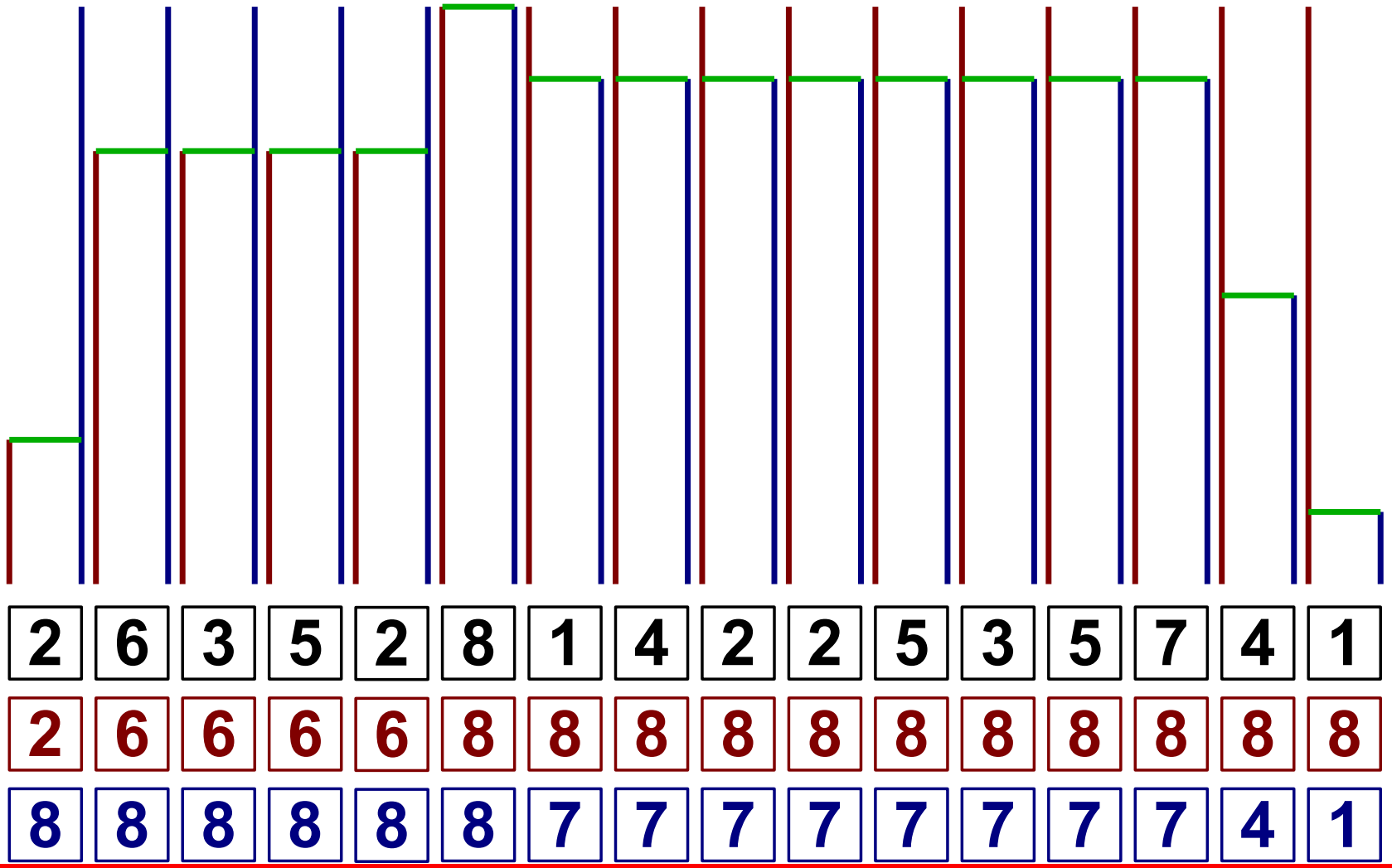
Left-to-Right Sweep



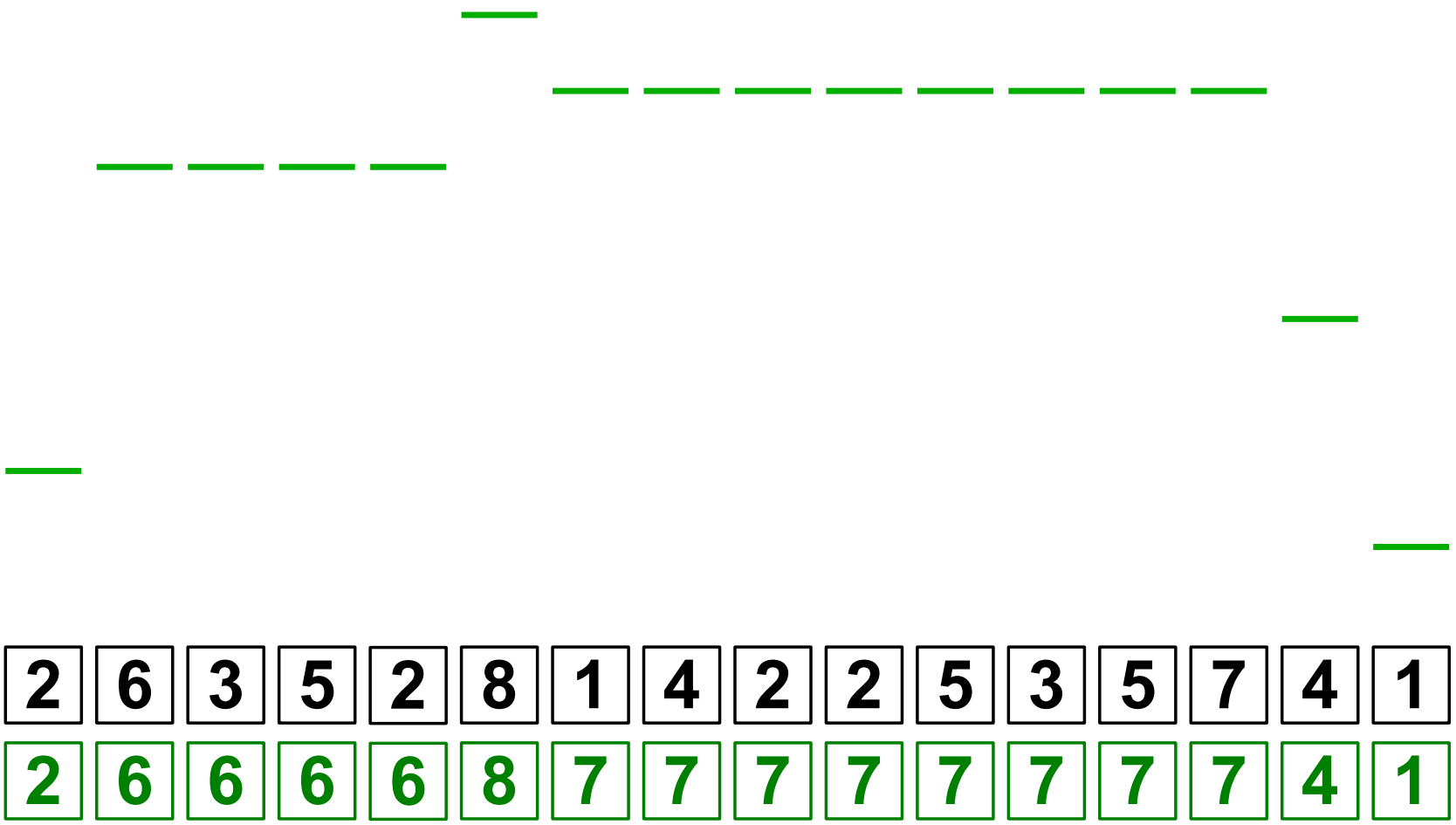
Right-to-Left Sweep



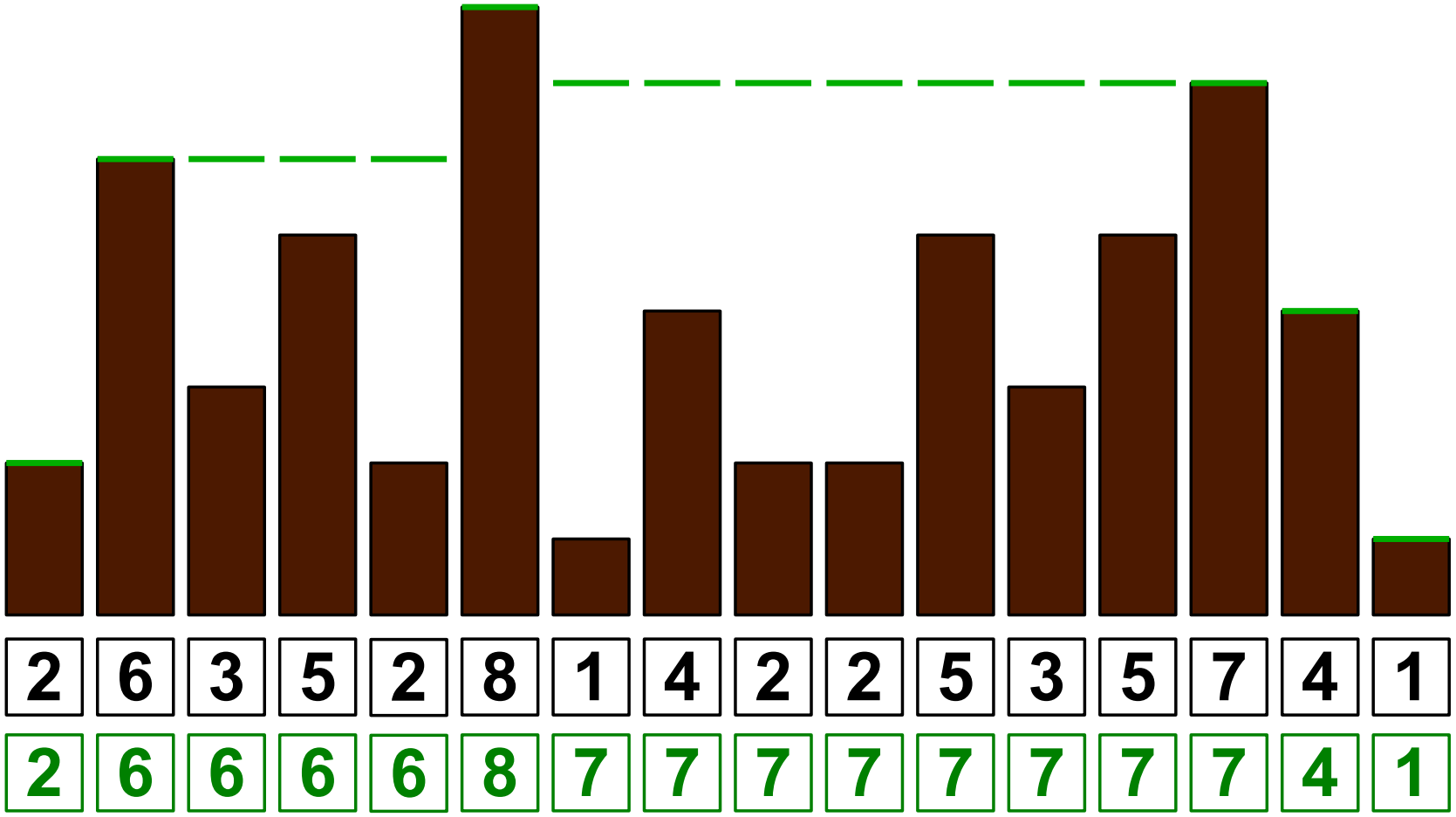
Find the Minimum (1 of 2)



Find the Minimum (2 of 2)

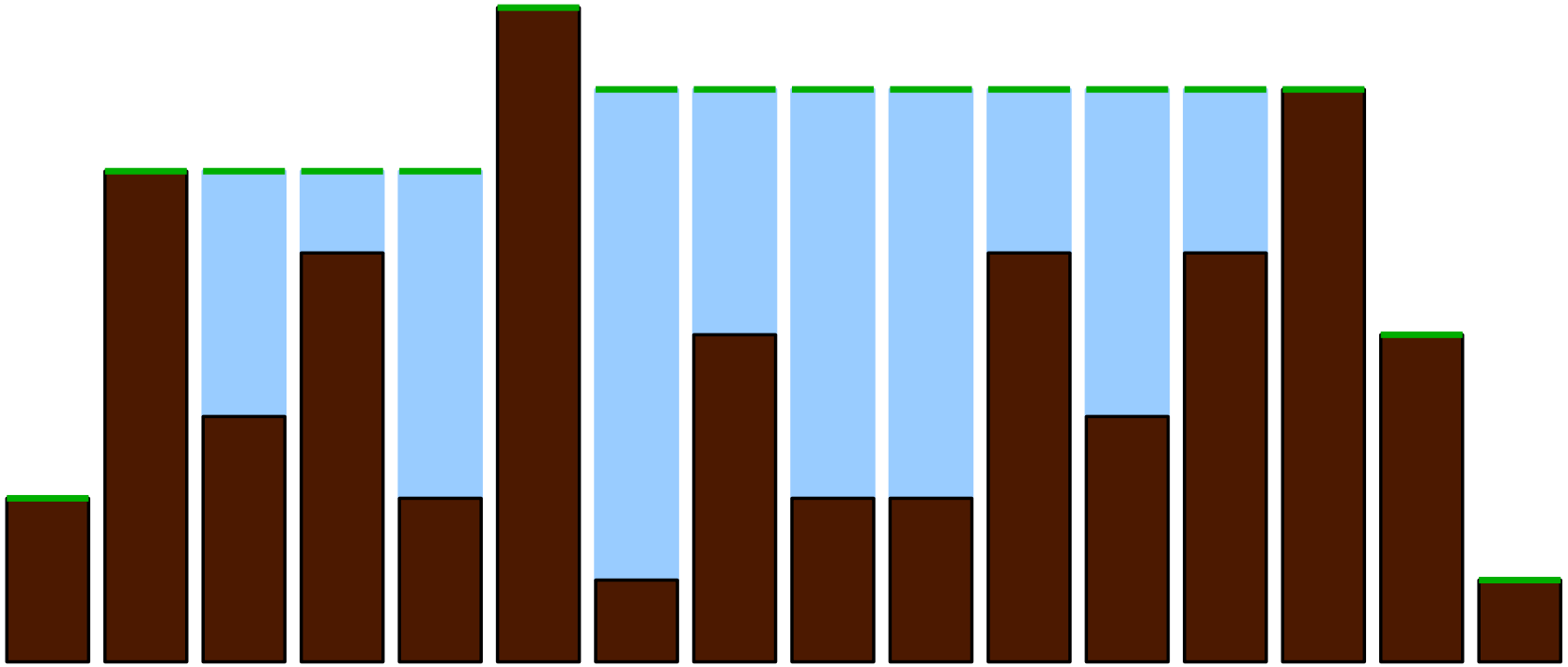


Compute Water for Each Bar (1 of 2)



Compute Water for Each Bar (2 of 2)

0	0	3	1	4	0	6	3	5	5	2	4	2	0	0	0



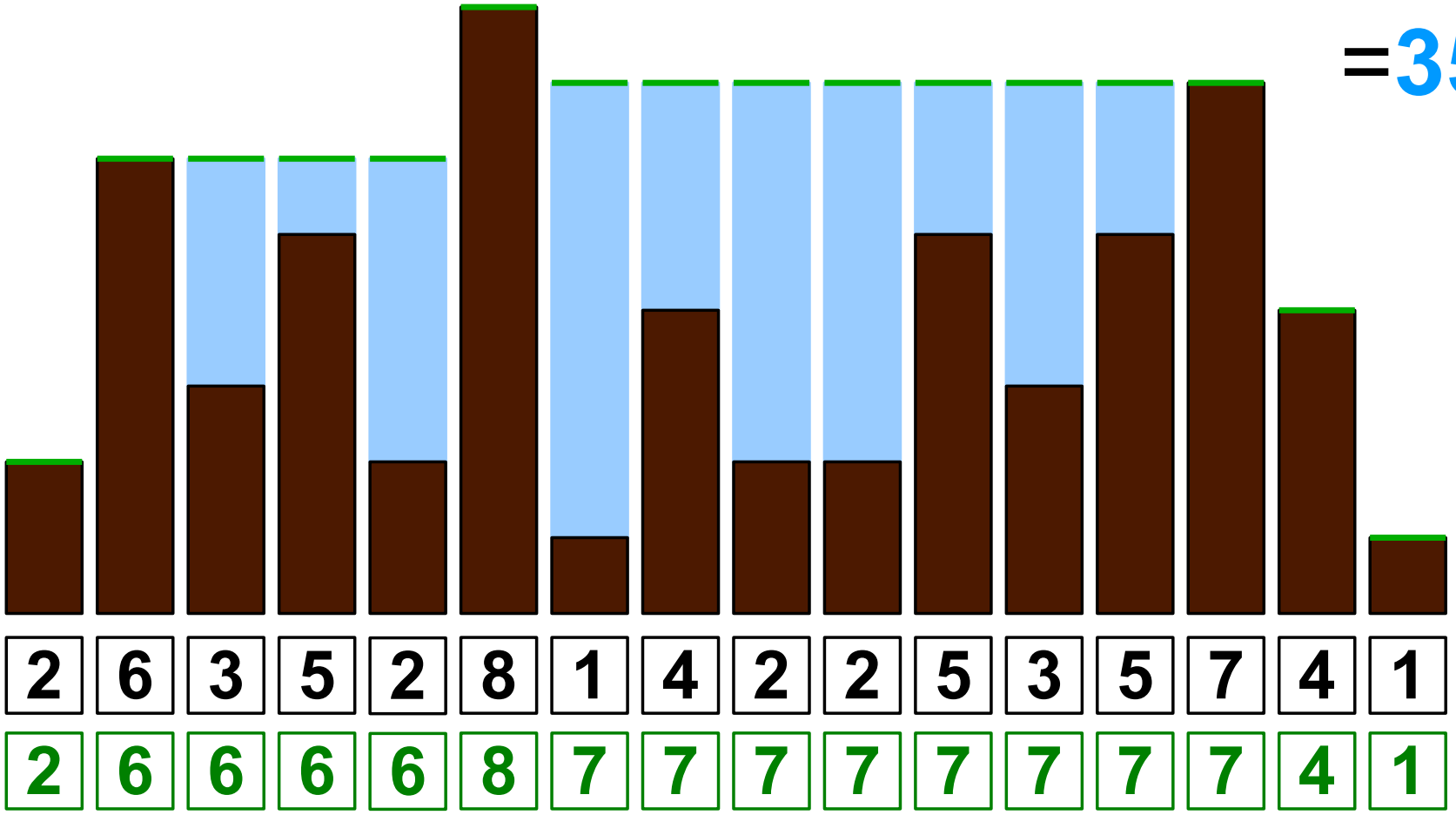
2	6	3	5	2	8	1	4	2	2	5	3	5	7	4	1

2	6	6	6	6	8	7	7	7	7	7	7	7	7	4	1

Compute Total Water (Summation)

$\sum$ 0 0 3 1 4 0 6 3 5 5 2 4 2 0 0 0

=35



Sequential Code

$$
\begin{array}{l}
\mathit{histogramWater}\big(x\colon \mathrm{Array}[\![\mathbb{Z}32, \mathbb{Z}32]\!]\big)\colon \mathbb{Z}32 = \texttt{do} \\
\quad n = |x| \\
\quad \texttt{if } n = 0 \texttt{ then } 0 \texttt{ else} \\
\qquad \mathit{left} = \mathit{array}[\![\mathbb{Z}32]\!](n) \\
\qquad \mathit{left}_0 := x_0 \\
\qquad \texttt{for } k \leftarrow \mathit{seq}(1 : n-1) \texttt{ do } \mathit{left}_k := \mathit{left}_{k-1} \texttt{ MAX } x_k \texttt{ end} \\
\qquad \mathit{right} = \mathit{array}[\![\mathbb{Z}32]\!](n) \\
\qquad \mathit{right}_{n-1} := x_{n-1} \\
\qquad \texttt{for } k \leftarrow \mathit{seq}(n-2 : 0 : -1) \texttt{ do } \mathit{right}_k := \mathit{right}_{k+1} \texttt{ MAX } x_k \texttt{ end} \\
\qquad \mathit{result}\colon \mathbb{Z}32 := 0 \\
\qquad \texttt{for } k \leftarrow \mathit{seq}(0 : n-1) \texttt{ do } \mathit{result} \mathrel{+}= \big((\mathit{left}_k \texttt{ MIN } \mathit{right}_k) - x_k\big) \texttt{ end} \\
\qquad \mathit{result} \\
\quad \texttt{end} \\
\texttt{end}
\end{array}
$$

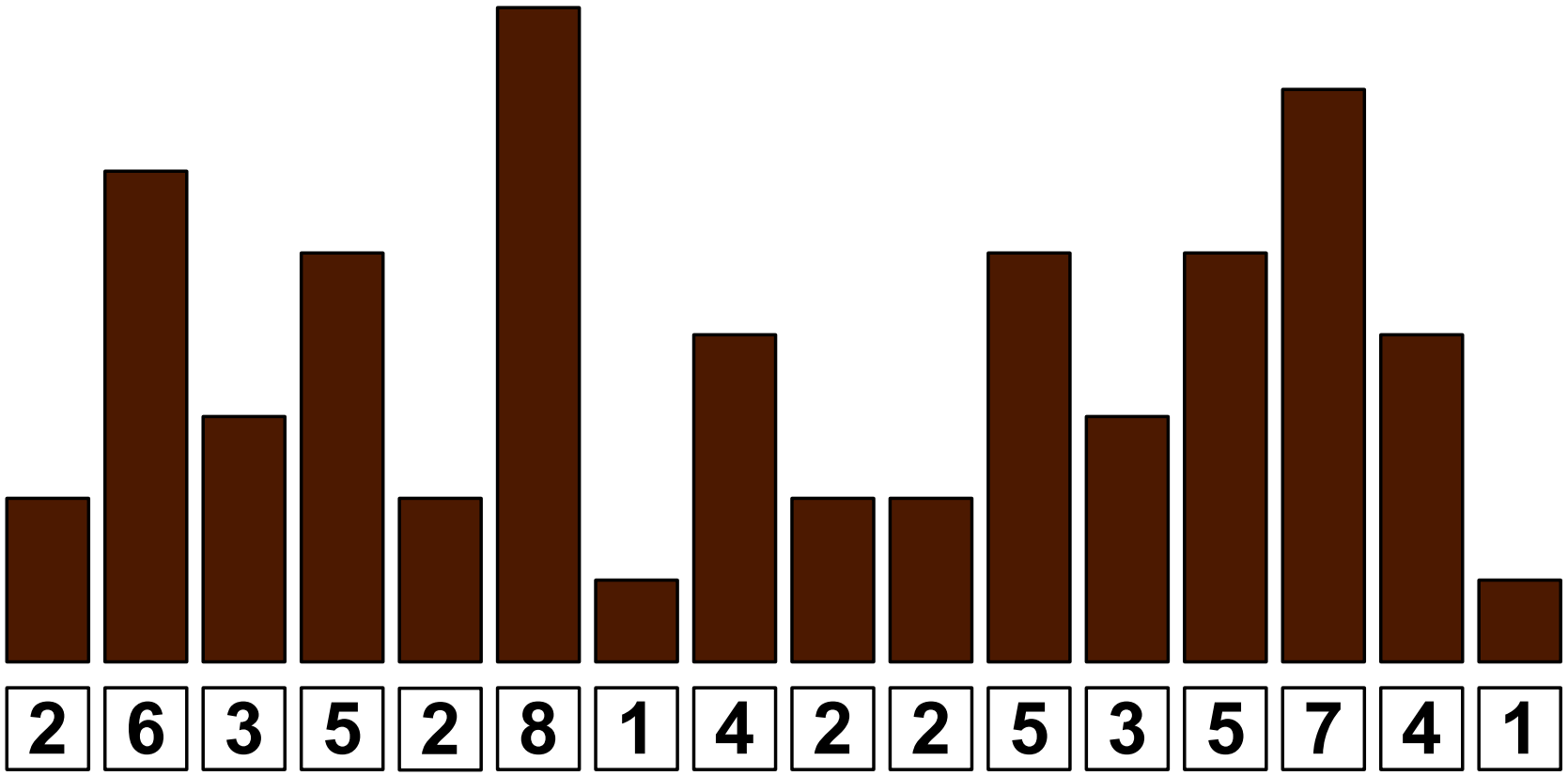
Sequential Code

$$
\begin{array}{l}
\mathit{histogramWater}\left(x\colon \mathrm{Array}[\![\mathbb{Z}32, \mathbb{Z}32]\!]\right)\colon \mathbb{Z}32 = \texttt{do} \\
\quad n = |x| \\
\quad \texttt{if } n = 0 \texttt{ then } 0 \texttt{ else} \\
\qquad \mathit{left} = \mathit{array}[\![\mathbb{Z}32]\!](n) \\
\qquad \mathit{left}_0 := x_0 \\
\qquad \texttt{for } k \leftarrow \mathit{seq}(1 : n-1) \texttt{ do } \mathit{left}_k := \mathit{left}_{k-1} \texttt{ MAX } x_k \texttt{ end} \\
\qquad \mathit{right} = \mathit{array}[\![\mathbb{Z}32]\!](n) \\
\qquad \mathit{right}_{n-1} := x_{n-1} \\
\qquad \texttt{for } k \leftarrow \mathit{seq}(n-2 : 0 : -1) \texttt{ do } \mathit{right}_k := \mathit{right}_{k+1} \texttt{ MAX } x_k \texttt{ end} \\
\qquad \mathit{result}\colon \mathbb{Z}32 := 0 \\
\qquad \texttt{for } k \leftarrow \mathit{seq}(0 : n-1) \texttt{ do } \mathit{result} \mathrel{+}= \left(\left(\mathit{left}_k \texttt{ MIN } \mathit{right}_k\right) - x_k\right) \texttt{ end} \\
\qquad \mathit{result} \\
\quad \texttt{end} \\
\texttt{end}
\end{array}
$$

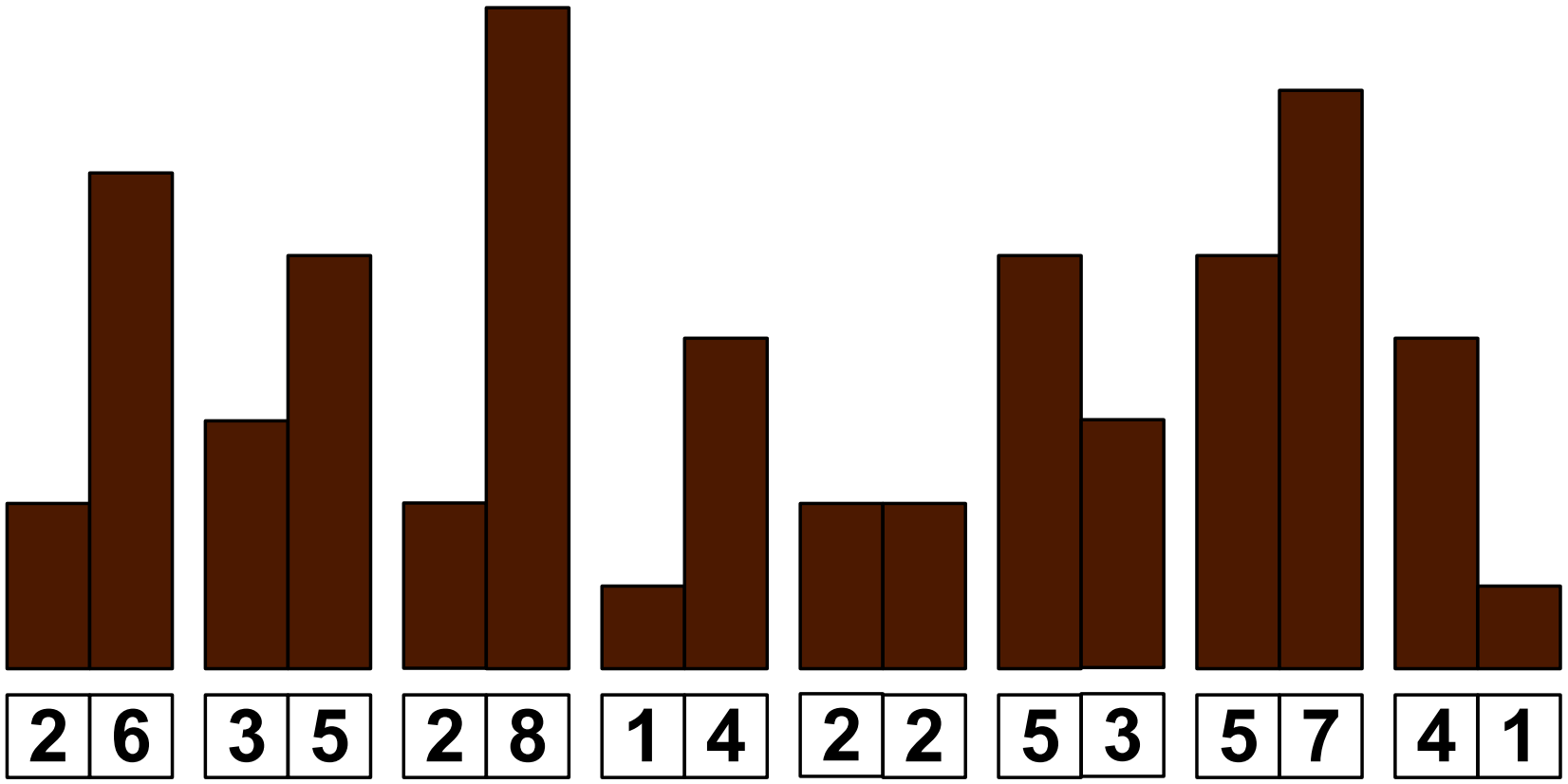
Sequential Code (Optimized)

```
histogramWater(x: Array[[ℤ32, ℤ32]]): ℤ32 = do
  n = |x|
  if n = 0 then 0 else
    left = array[[ℤ32]](n)
    left_0 := x_0
    for k ← seq(1 : n − 1) do left_k := left_{k−1} MAX x_k end
    right: ℤ32 := x_{n−1}
    result: ℤ32 := 0
    for k ← seq(n − 1 : 0 : − 1) do
      right MAX= x_k
      result += ((left_k MIN right) − x_k)
    end
    result
  end
end
```

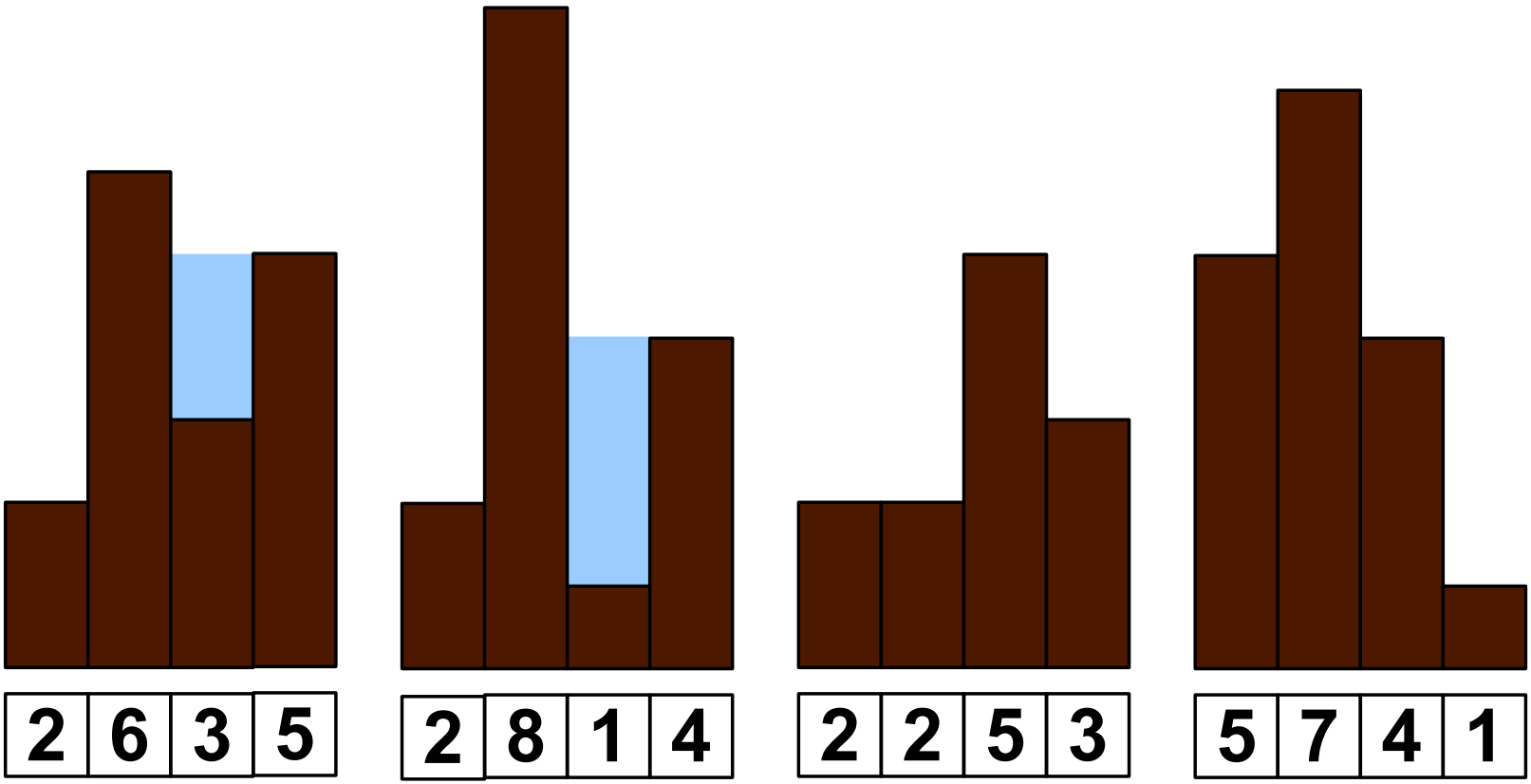
Divide and Conquer (1 of 5)



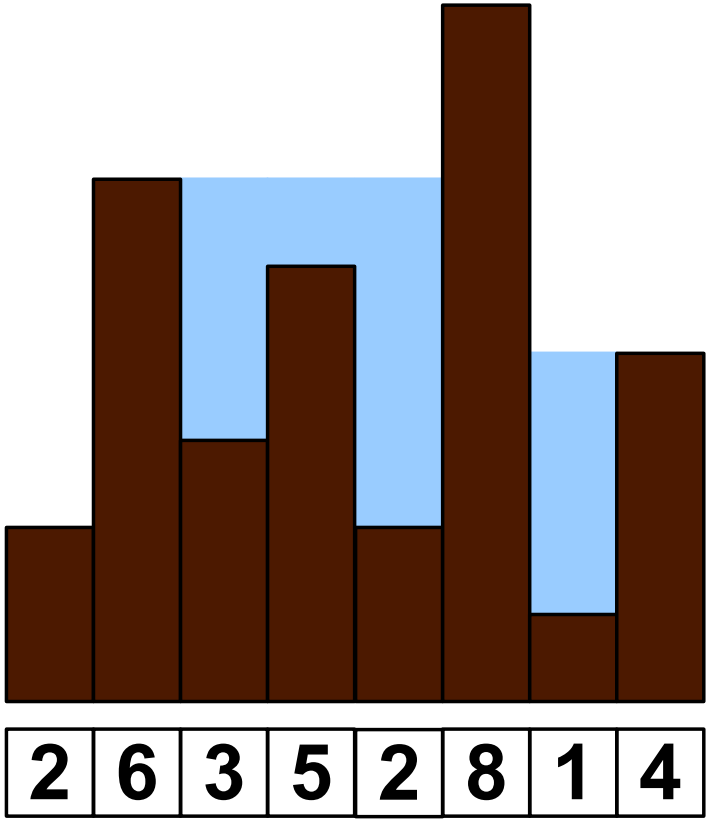
Divide and Conquer (2 of 5)

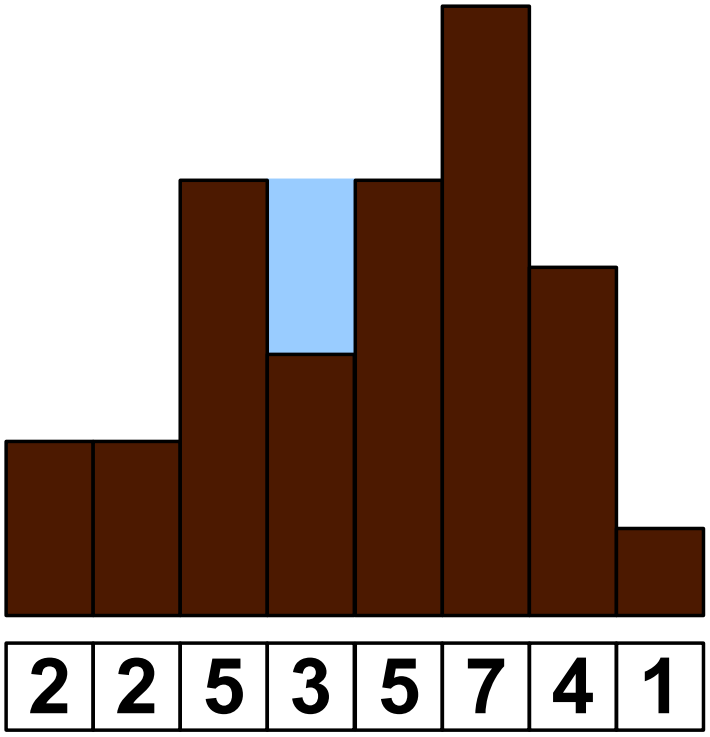


Divide and Conquer (3 of 5)

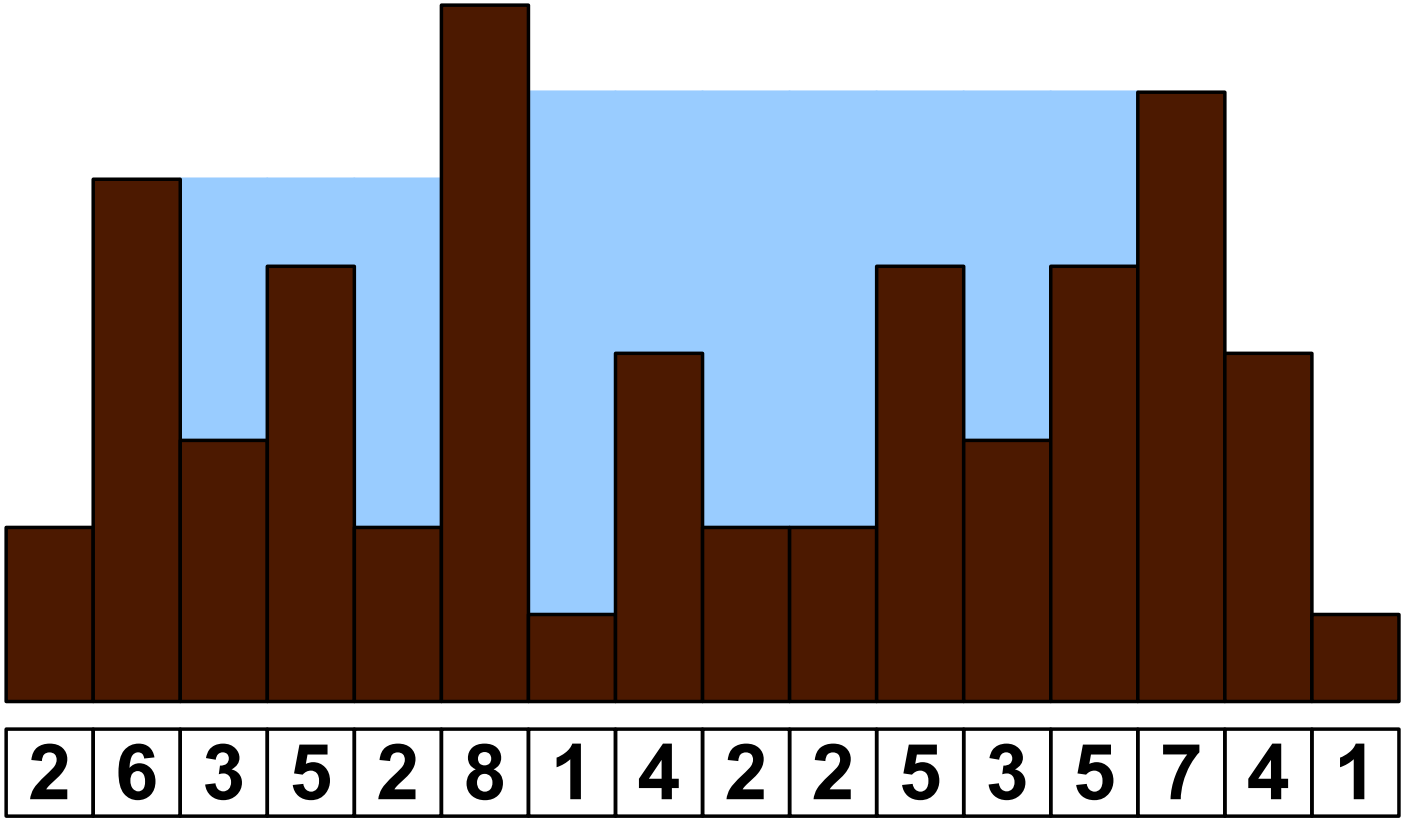


Divide and Conquer (4 of 5)

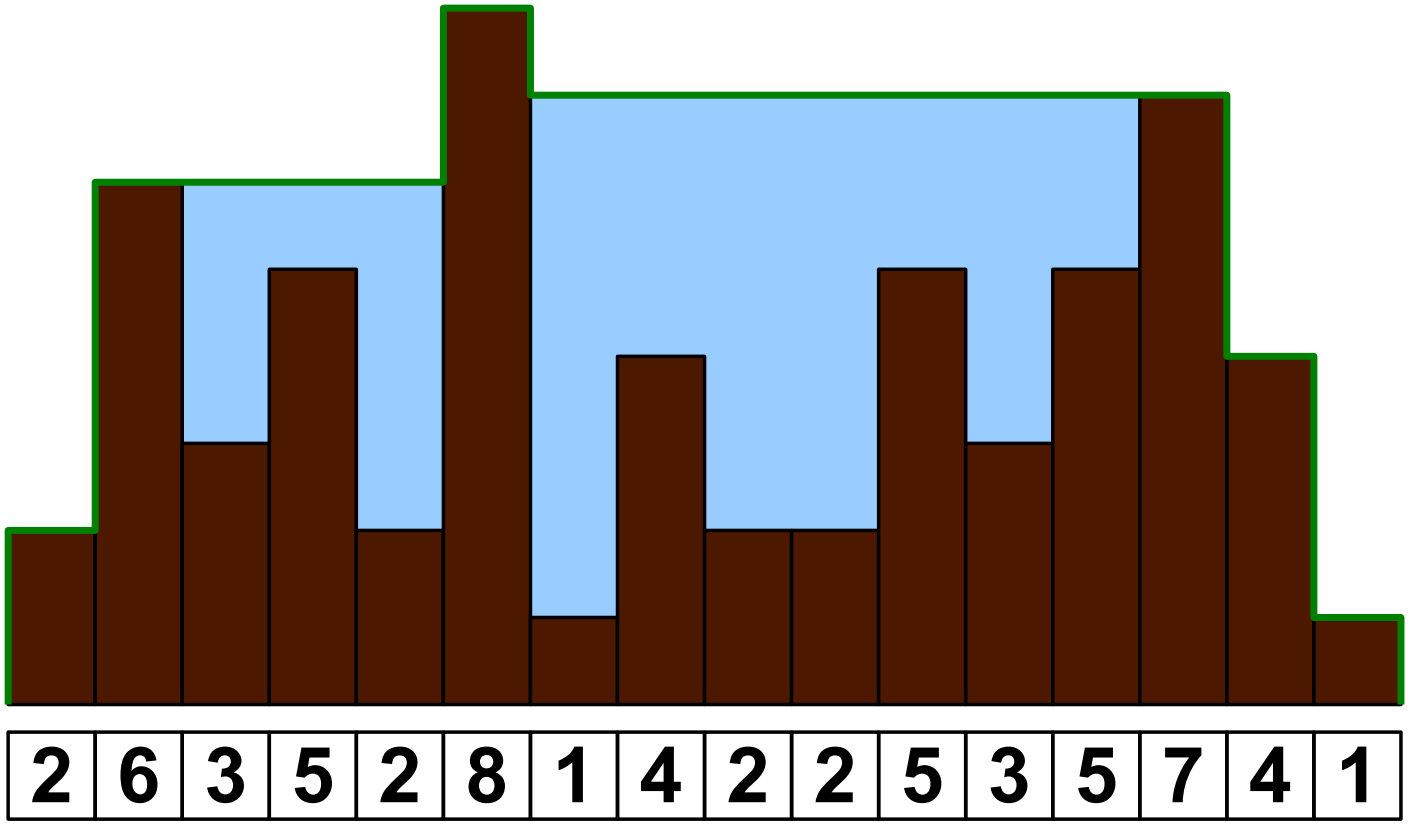




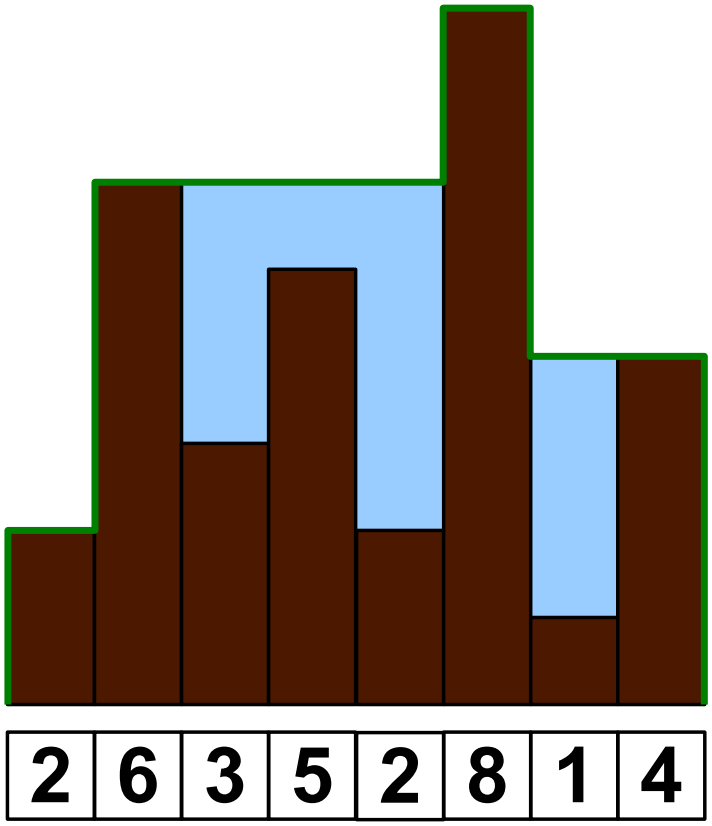
Divide and Conquer (5 of 5)

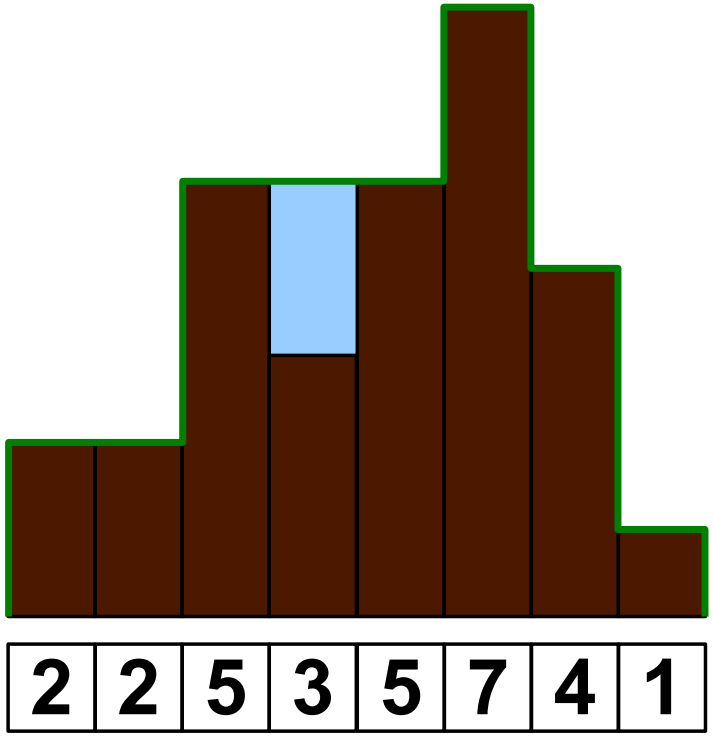


Bitonic Outline

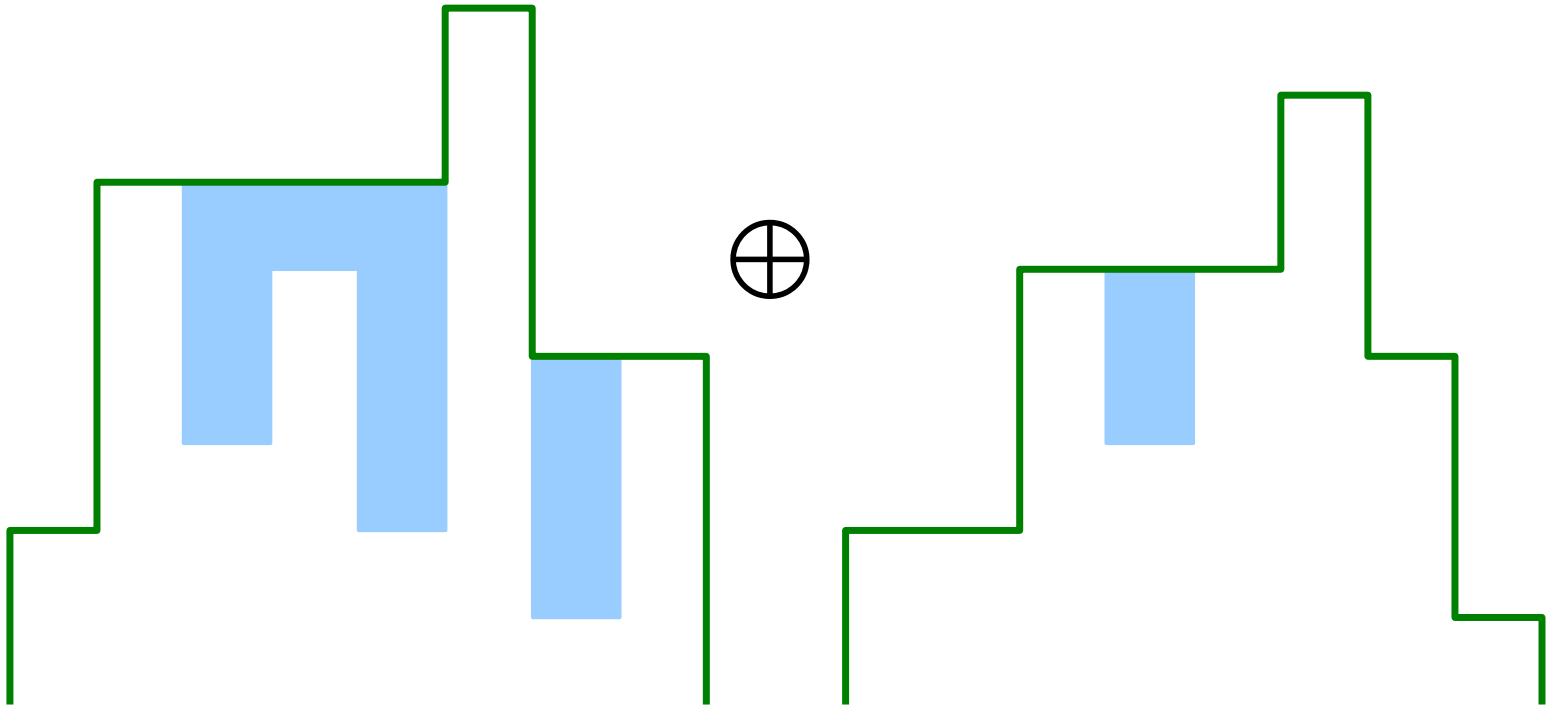


Combining Two Bitonic Globs (1 of 8)

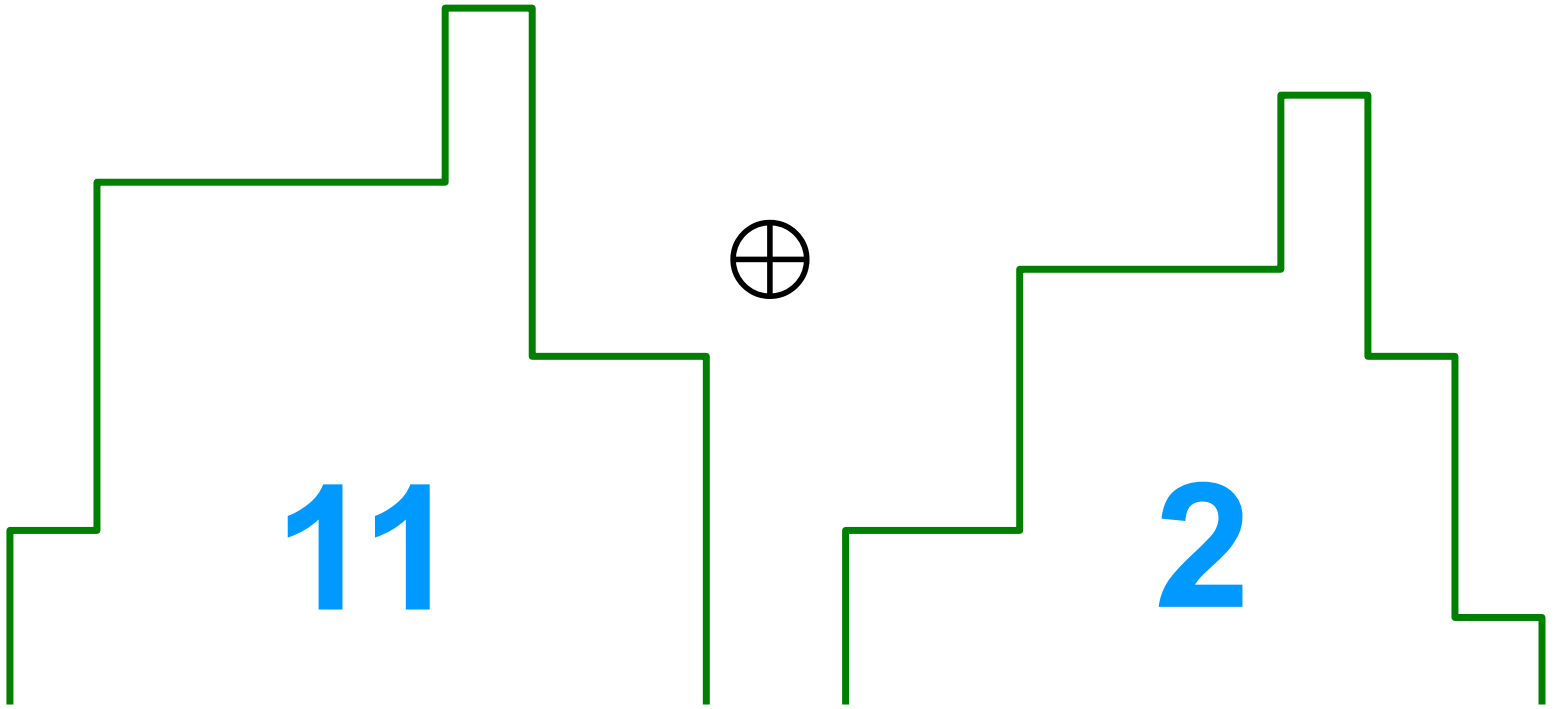




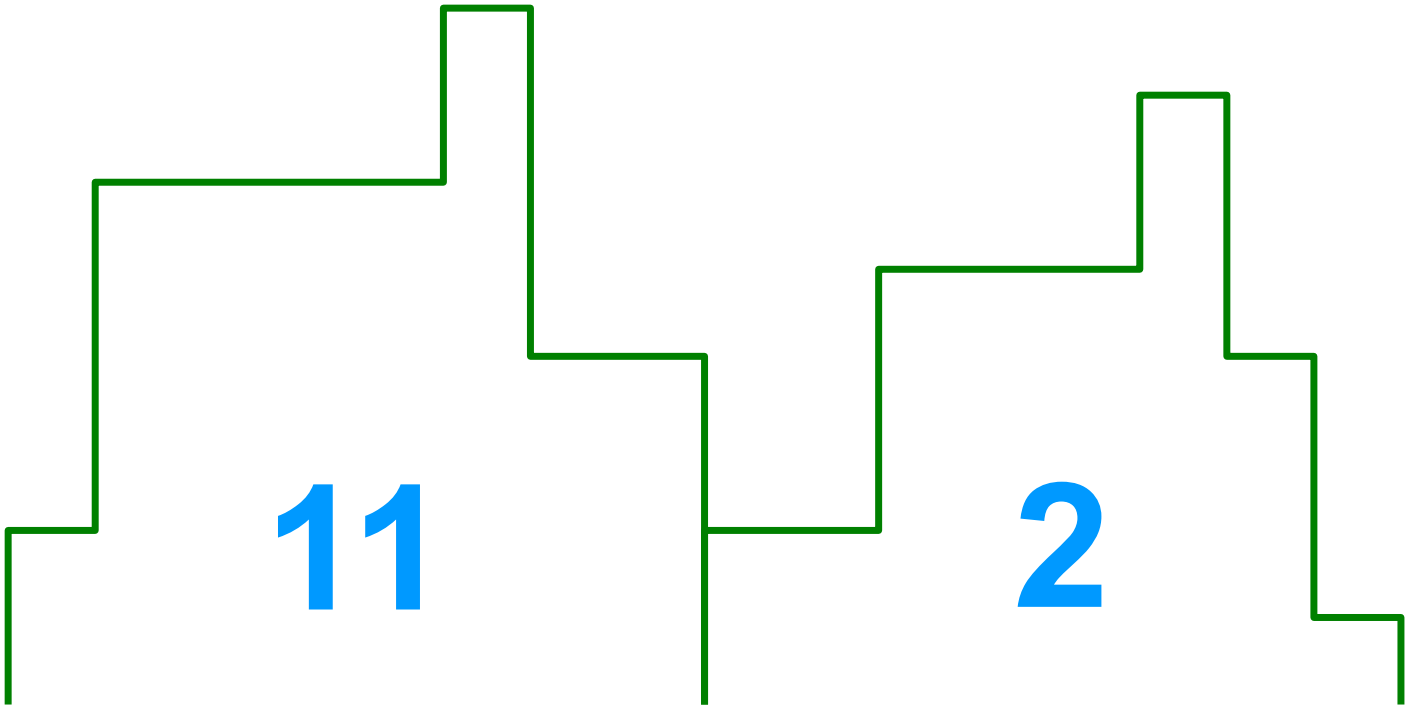
Combining Two Bitonic Globs (2 of 8)



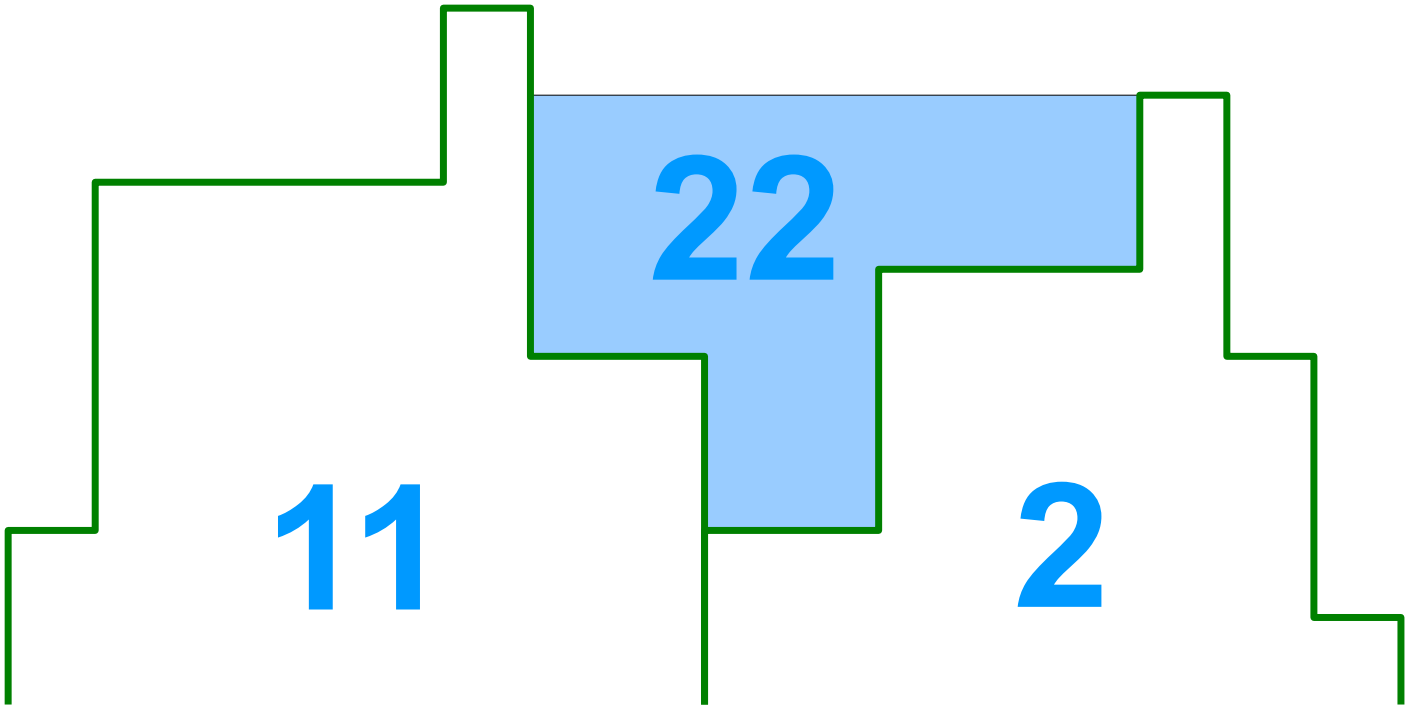
Combining Two Bitonic Globs (3 of 8)



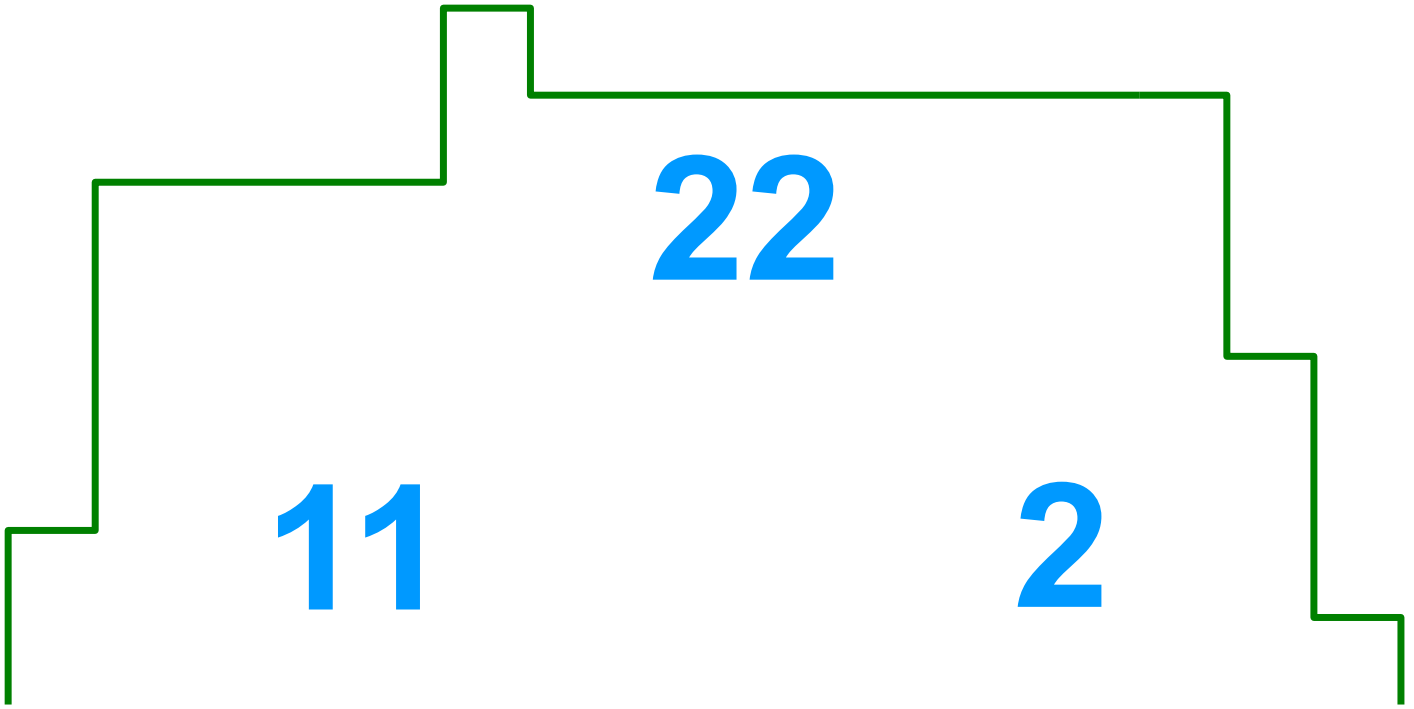
Combining Two Bitonic Globs (4 of 8)



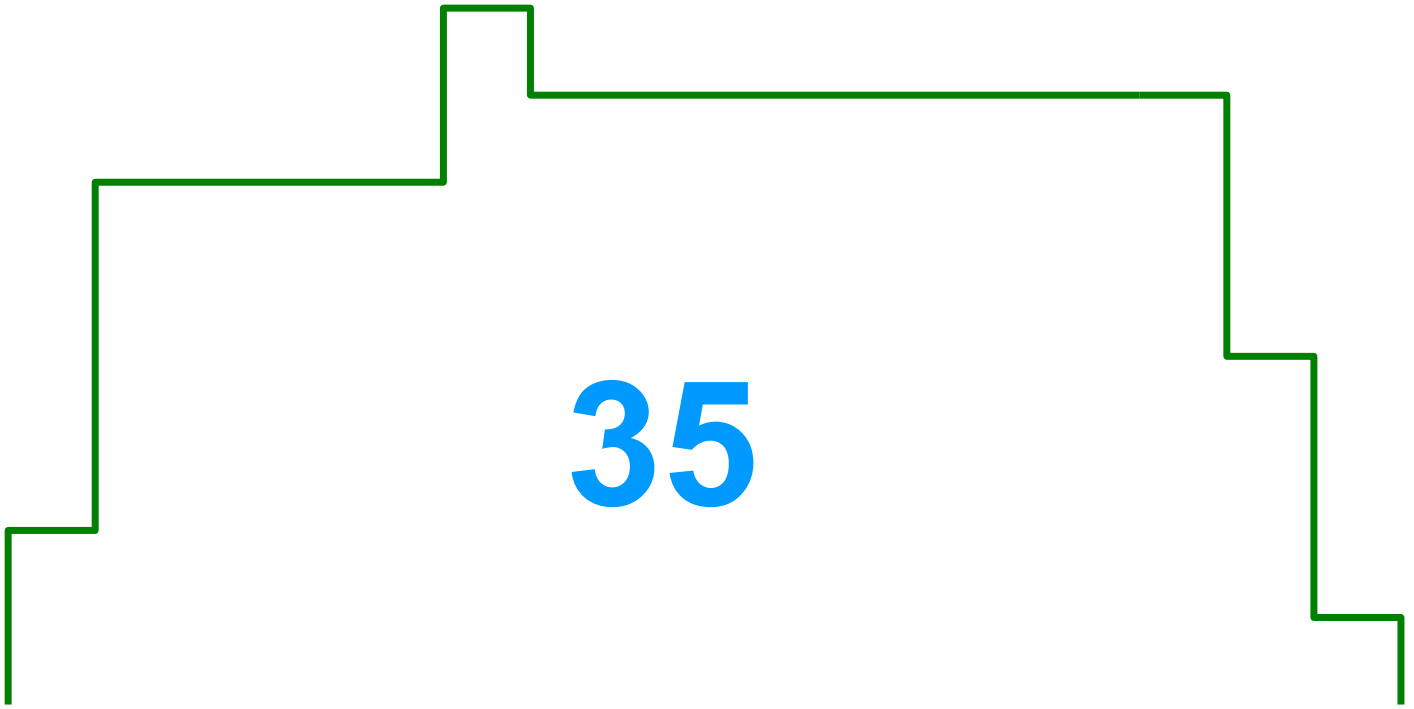
Combining Two Bitonic Globs (5 of 8)



Combining Two Bitonic Globs (6 of 8)



Combining Two Bitonic Globs (7 of 8)



Combining Two Bitonic Globs (8 of 8)

$$
\begin{array}{l}
\texttt{opr}\ \oplus(x\colon \mathrm{Glob}, y\colon \mathrm{Glob})\colon \mathrm{Glob} = \\
\quad \texttt{case}\ (x.ht\ \texttt{CMP}\ y.ht)\ \texttt{of} \\
\qquad \mathrm{LessThan} \Rightarrow \\
\qquad\quad (lss, eql, gtr) = threeWaySplit(y.left, x.ht) \\
\qquad\quad \mathrm{Glob}\big(x.left \parallel \langle\, (x.ht, x.wd + width(x.right) + width(lss) + eql.getDefault(0))\,\rangle \parallel gtr, \\
\qquad\qquad\quad y.ht, y.wd, y.right, \\
\qquad\qquad\quad x.water + fill(x.right, x.ht) + fill(lss, x.ht) + y.water\big) \\
\qquad \mathrm{GreaterThan} \Rightarrow \\
\qquad\quad (lss, eql, gtr) = threeWaySplit(x.right, y.ht) \\
\qquad\quad \mathrm{Glob}(x.left, x.ht, x.wd, \\
\qquad\qquad\quad y.right \parallel \langle\, (y.ht, eql.getDefault(0) + width(lss) + width(y.left) + y.wd)\,\rangle \parallel gtr, \\
\qquad\qquad\quad x.water + fill(lss, y.ht) + fill(y.left, y.ht) + y.water) \\
\qquad \mathrm{EqualTo} \Rightarrow \\
\qquad\quad \mathrm{Glob}(x.left, x.ht, x.wd + width(x.right) + width(y.left) + y.wd, y.right, \\
\qquad\qquad\quad x.water + fill(x.right, x.ht) + fill(y.left, x.ht) + y.water) \\
\quad \texttt{end}
\end{array}
$$

Is it associative?

Glob Data Structure and Utility Operations

$$\texttt{object}\ \mathrm{Glob}\big(\mathit{left}\colon \mathrm{List}[\![(\mathbb{Z}32, \mathbb{Z}32)]\!],\ \mathit{ht}\colon \mathbb{Z}32,\ \mathit{wd}\colon \mathbb{Z}32,\ \mathit{right}\colon \mathrm{List}[\![(\mathbb{Z}32, \mathbb{Z}32)]\!],\ \mathit{water}\colon \mathbb{Z}32\big)$$

$$\texttt{end}$$

$$\mathit{width}\big(x\colon \mathrm{List}[\![(\mathbb{Z}32, \mathbb{Z}32)]\!]\big) = \sum_{(p,q)\leftarrow x} q$$

$$\mathit{fill}\big(x\colon \mathrm{List}[\![(\mathbb{Z}32, \mathbb{Z}32)]\!],\ m\colon \mathbb{Z}32\big) = \sum_{(p,q)\leftarrow x} q \times (m - p)$$

$$\begin{aligned}
&\texttt{object}\ \mathrm{GlobReduction}\\
&\qquad \texttt{extends}\ \{\ \mathrm{CommutativeMonoidReduction}[\![\mathrm{Glob}]\!],\\
&\qquad\qquad \mathrm{ReductionWithZeroes}[\![\mathrm{Glob}, \mathrm{Glob}]\!]\ \}\\
&\quad \texttt{getter}\ \mathit{asString}()\colon \mathrm{String} = \text{“}\texttt{GlobReduction}\text{”}\\
&\quad \mathit{empty}()\colon \mathrm{Glob} = \mathrm{Glob}\big(\langle\,\rangle, -\infty, 0, \langle\,\rangle, 0\big)\\
&\quad \mathit{join}(a\colon \mathrm{Glob}, b\colon \mathrm{Glob})\colon \mathrm{Glob} = a \oplus b\\
&\texttt{end}
\end{aligned}$$

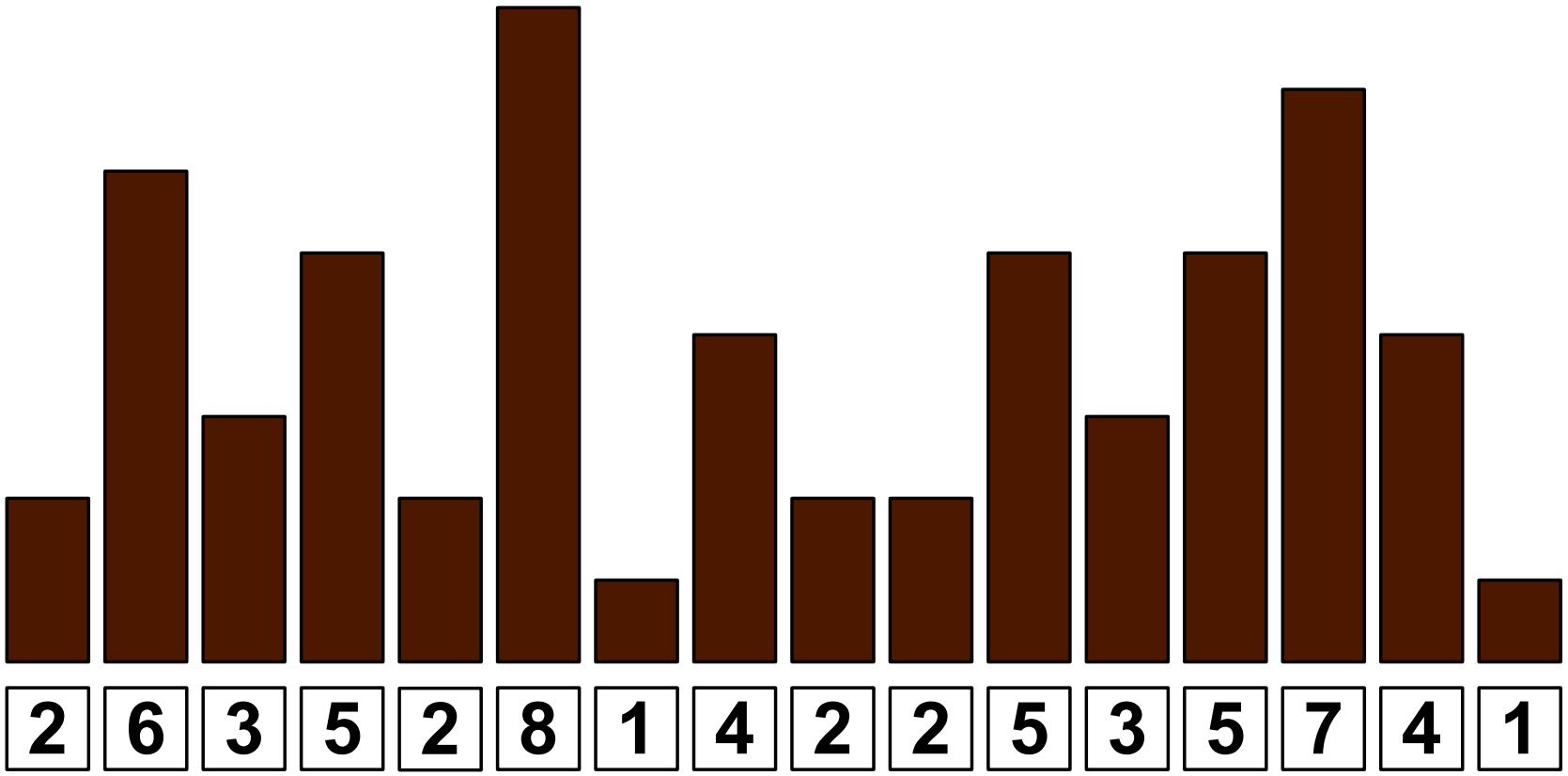
$$\texttt{opr BIG}\ \oplus()\colon \mathrm{BigReduction}[\![\mathrm{Glob}, \mathrm{Glob}]\!] = \mathrm{BigReduction}[\![\mathrm{Glob}, \mathrm{Glob}]\!](\mathrm{GlobReduction})$$

Utility Operation for Splitting Sorted Lists

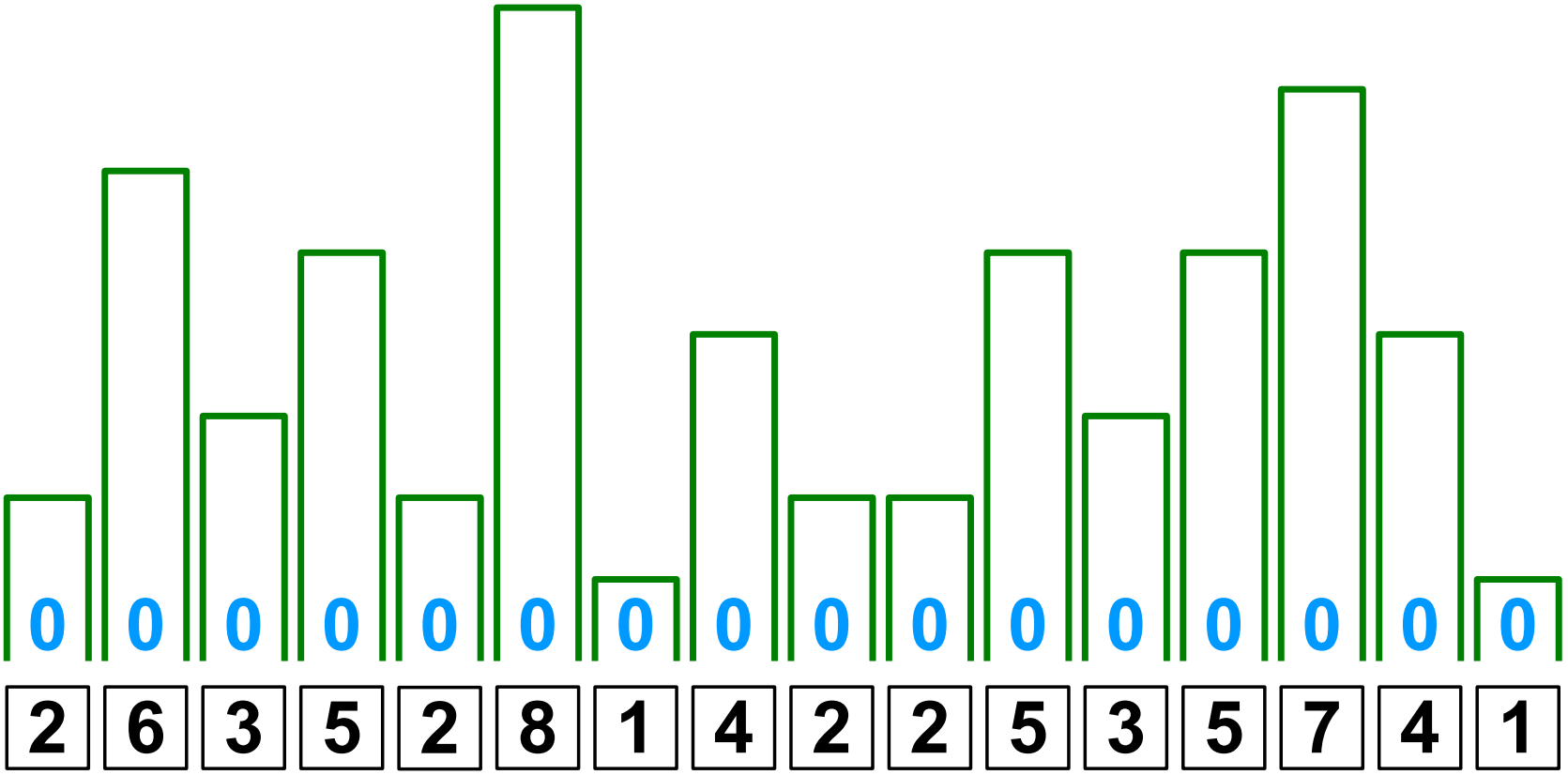
$threeWaySplit\big(x\colon \mathrm{List}[\![(\mathbb{Z}32, \mathbb{Z}32)]\!], m\colon \mathbb{Z}32\big)\colon \big(\mathrm{List}[\![(\mathbb{Z}32, \mathbb{Z}32)]\!], \mathrm{Maybe}[\![\mathbb{Z}32]\!], \mathrm{List}[\![(\mathbb{Z}32, \mathbb{Z}32)]\!]\big) =$

```
if |x| = 0 then (⟨⟩, Nothing, ⟨⟩)
elif |x| = 1 then
  (a, b) = x_0
  if a < m then (x, Nothing, ⟨⟩)
  elif a > m then (⟨⟩, Nothing, x)
  else (⟨⟩, Just b, ⟨⟩) end
else
  (y, z) = x.split()      ⊛ into approximately equal pieces
  (n, w) = z_0
  if m < n then
    (p, q, r) = threeWaySplit(y, m)
    (p, q, r ‖ z)
  else
    (p, q, r) = threeWaySplit(z, m)
    (y ‖ p, q, r)
  end
end
```

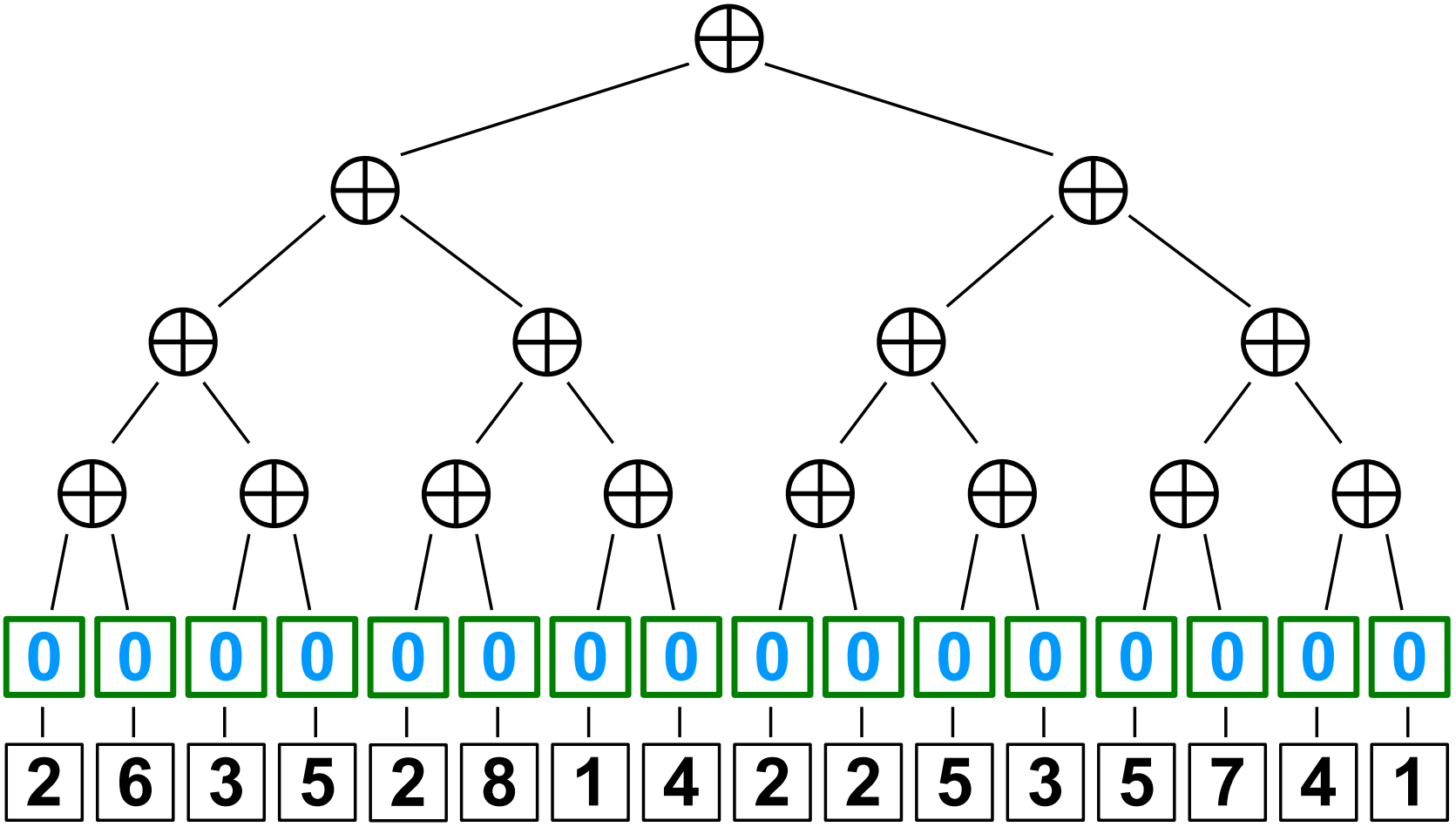
Singleton Bitonic Globs (1 of 2)



Singleton Bitonic Globs (2 of 2)

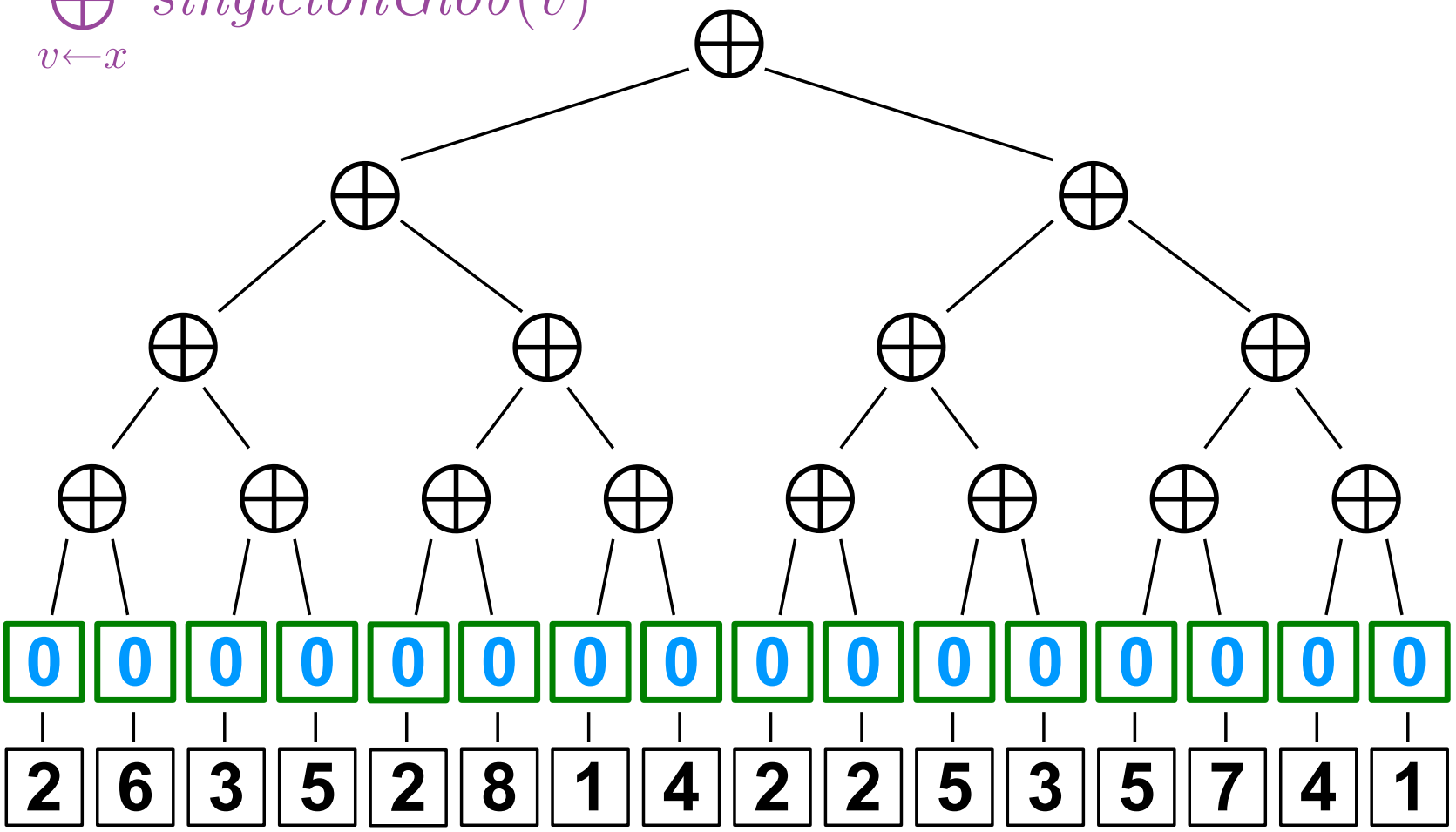


Combining Singleton Globs (1 of 4)

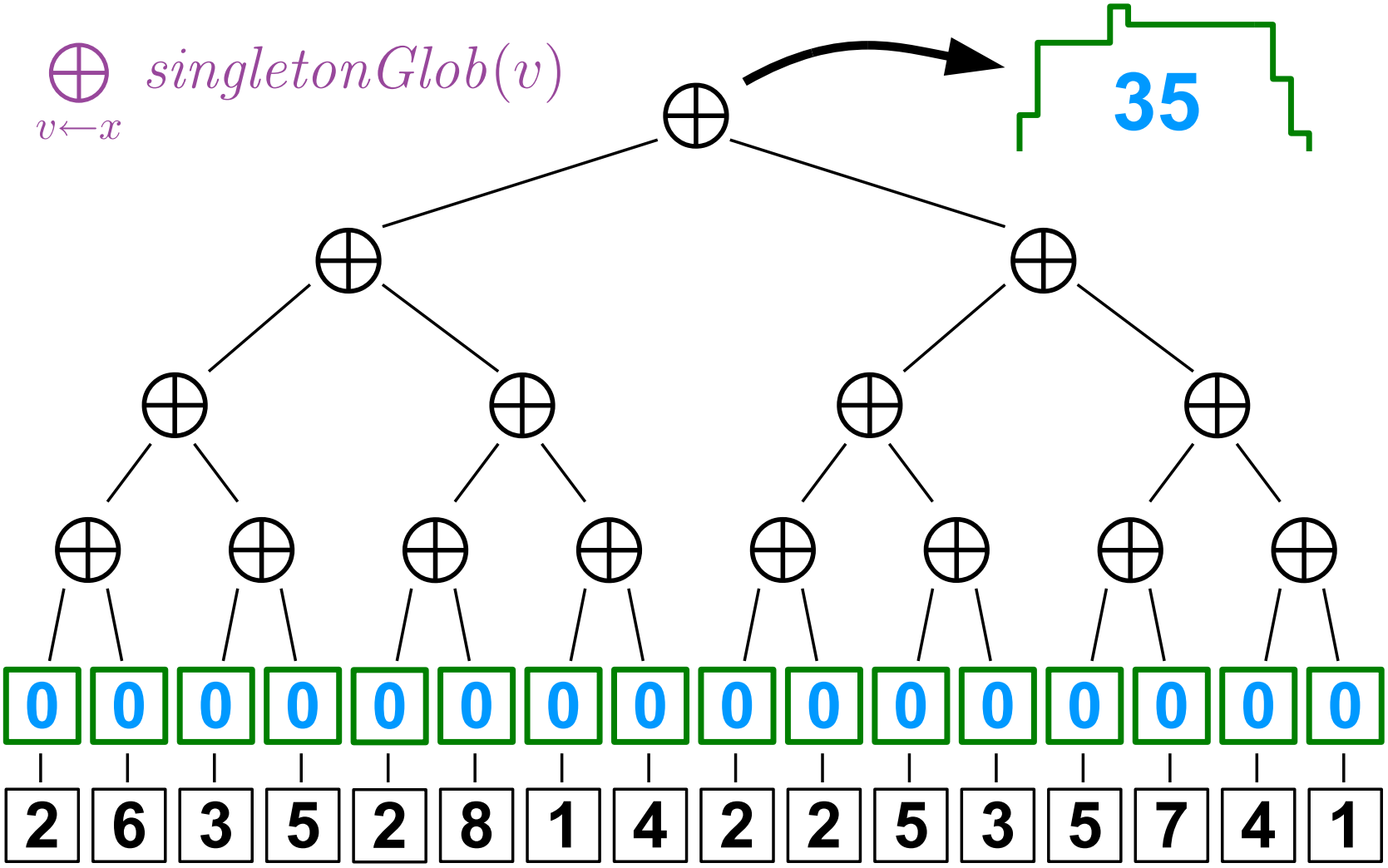


Combining Singleton Globs (2 of 4)

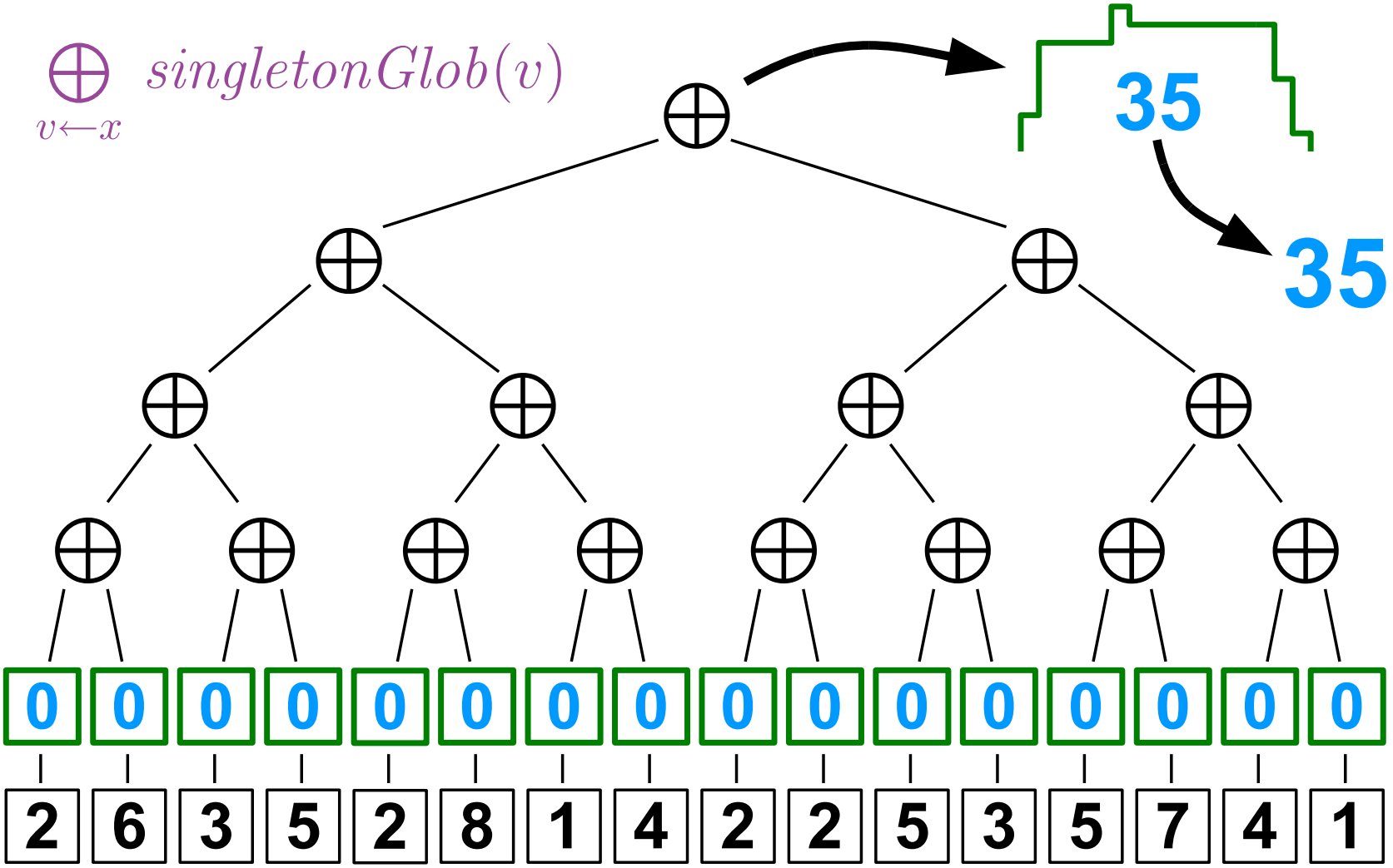
$\bigoplus_{v \leftarrow x} singletonGlob(v)$



Combining Singleton Globs (3 of 4)



Combining Singleton Globs (3 of 4)



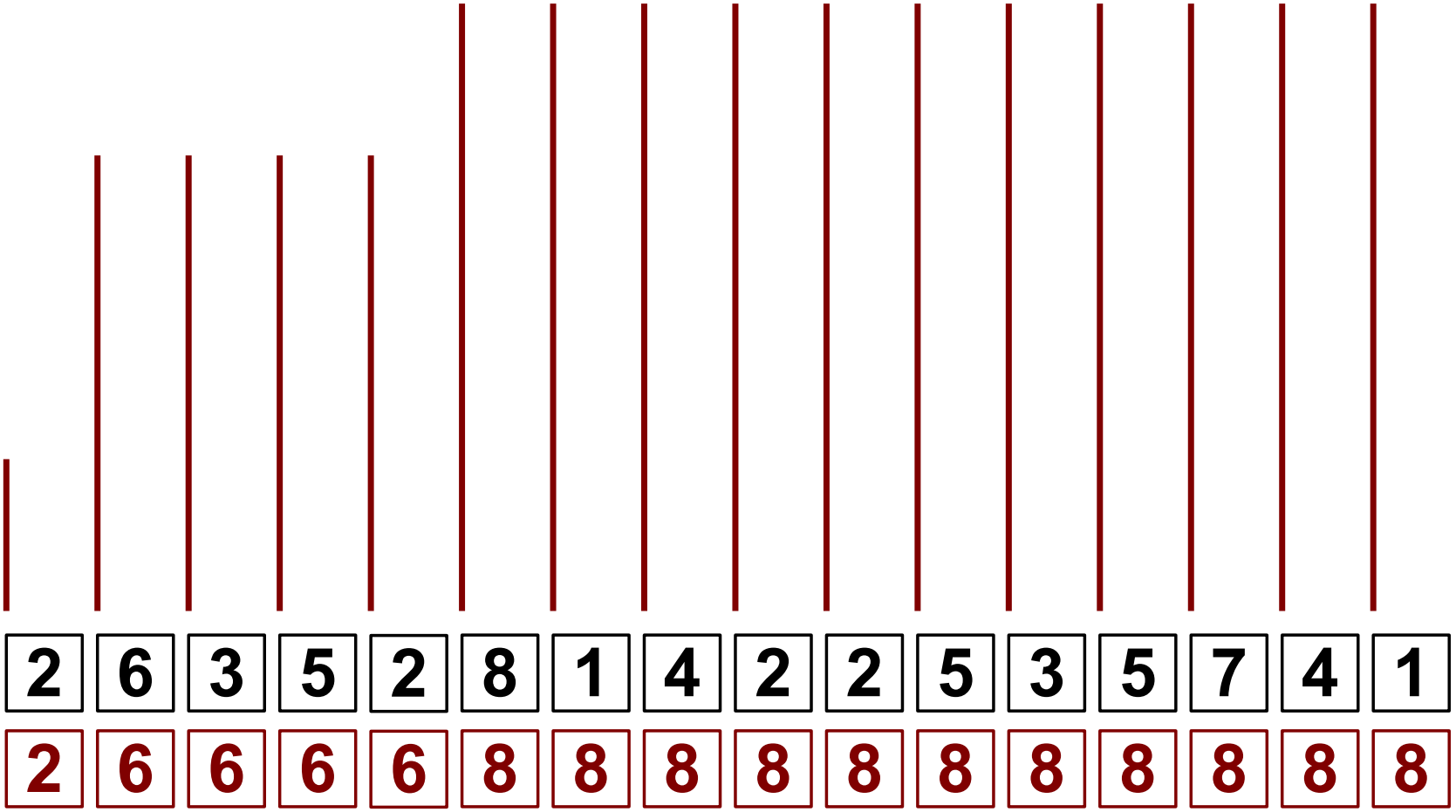
Final Divide-and-Conquer Code

$$histogramWater\big(x\colon \mathrm{List}[\![\mathbb{Z}32]\!]\big)\colon \mathbb{Z}32 = \left(\bigoplus_{v \leftarrow x} \mathrm{Glob}(\langle\,\rangle, v, 1, \langle\,\rangle, 0)\right).water$$

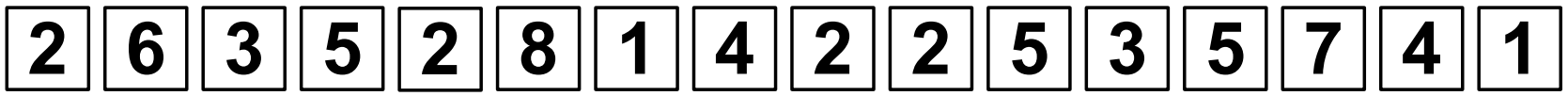
Final Divide-and-Conquer Code

$$histogramWater\big(x\colon \mathrm{List}[\![\mathbb{Z}32]\!]\big)\colon \mathbb{Z}32 = \left(\bigoplus_{v \leftarrow x} \mathrm{Glob}(\langle\,\rangle, v, 1, \langle\,\rangle, 0)\right).water$$

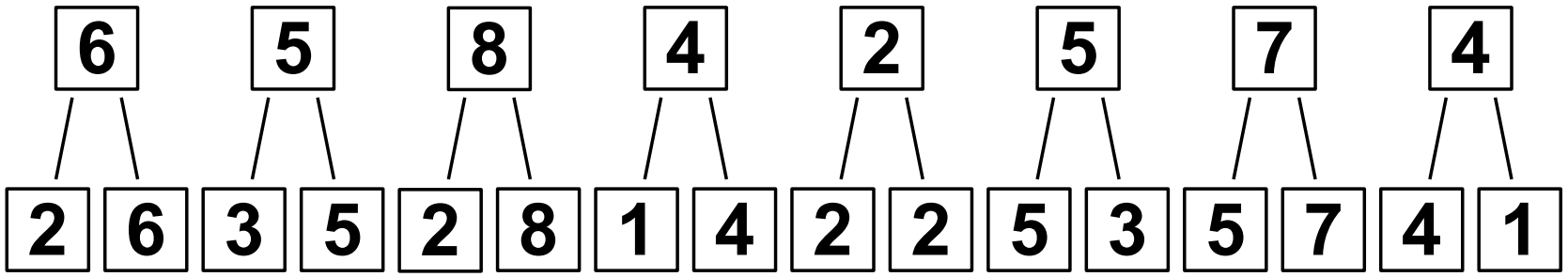
Left-to-Right Sweep



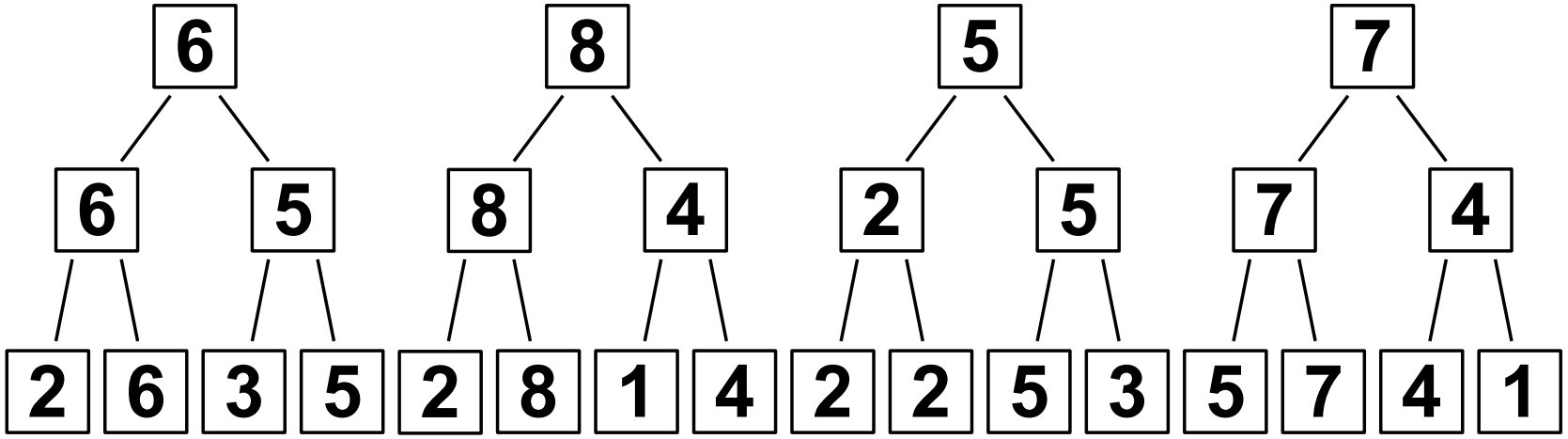
Monoid-cached Tree (1 of 5)



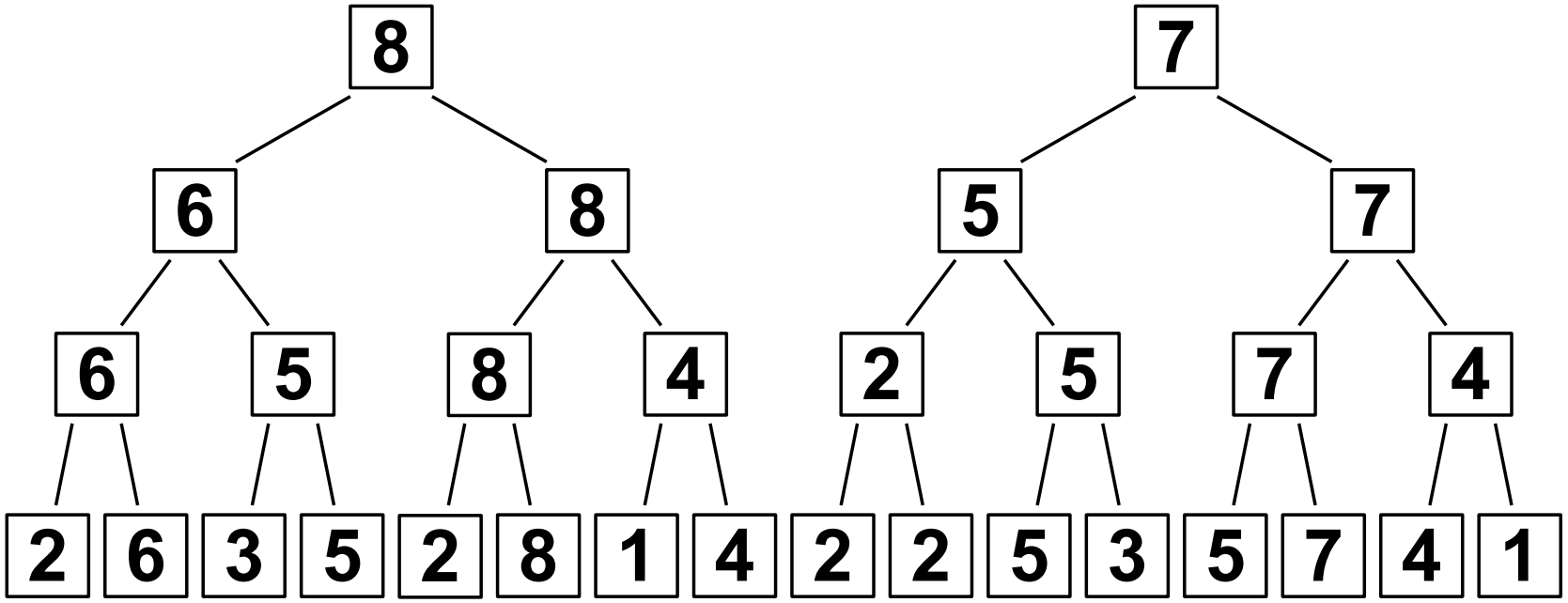
Monoid-cached Tree (2 of 5)



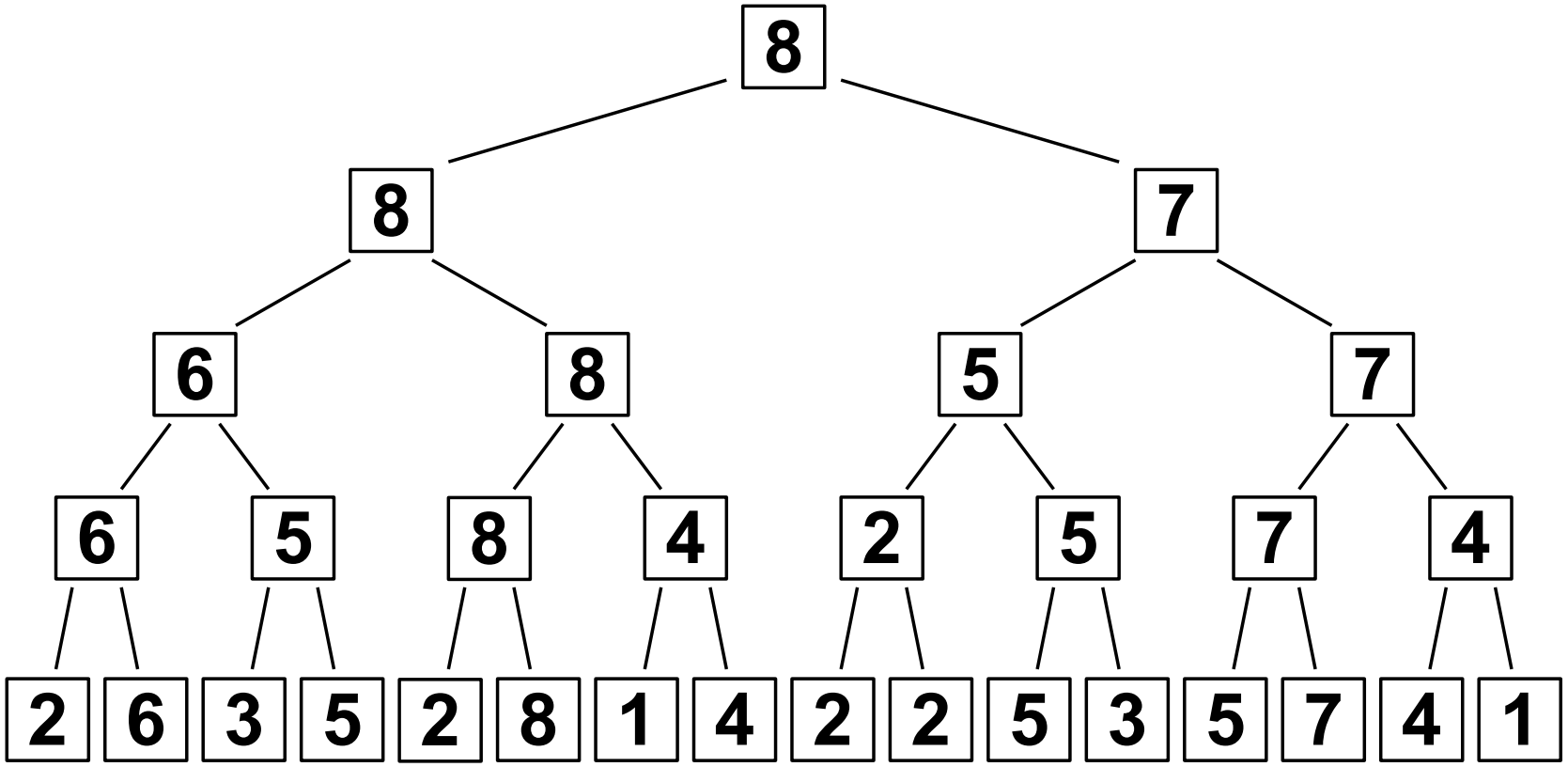
Monoid-cached Tree (3 of 5)



Monoid-cached Tree (4 of 5)



Monoid-cached Tree (5 of 5)



See Hinze, Ralf, and Paterson, Ross. “Finger Trees: A Simple General-purpose Data Structure.” *Journal of Functional Programming 16* (2): 2006, 197–217.

Code to Make a MAX-cached Tree (1 of 2)

```
trait CachedTree comprises { NullNode, SingletonNode, PairNode }
  getter val(): ℤ32
end

object NullNode(val: ℤ32) extends CachedTree end
object SingletonNode(val: ℤ32) extends CachedTree end
object PairNode(val: ℤ32, a: CachedTree, b: CachedTree) extends CachedTree end
```

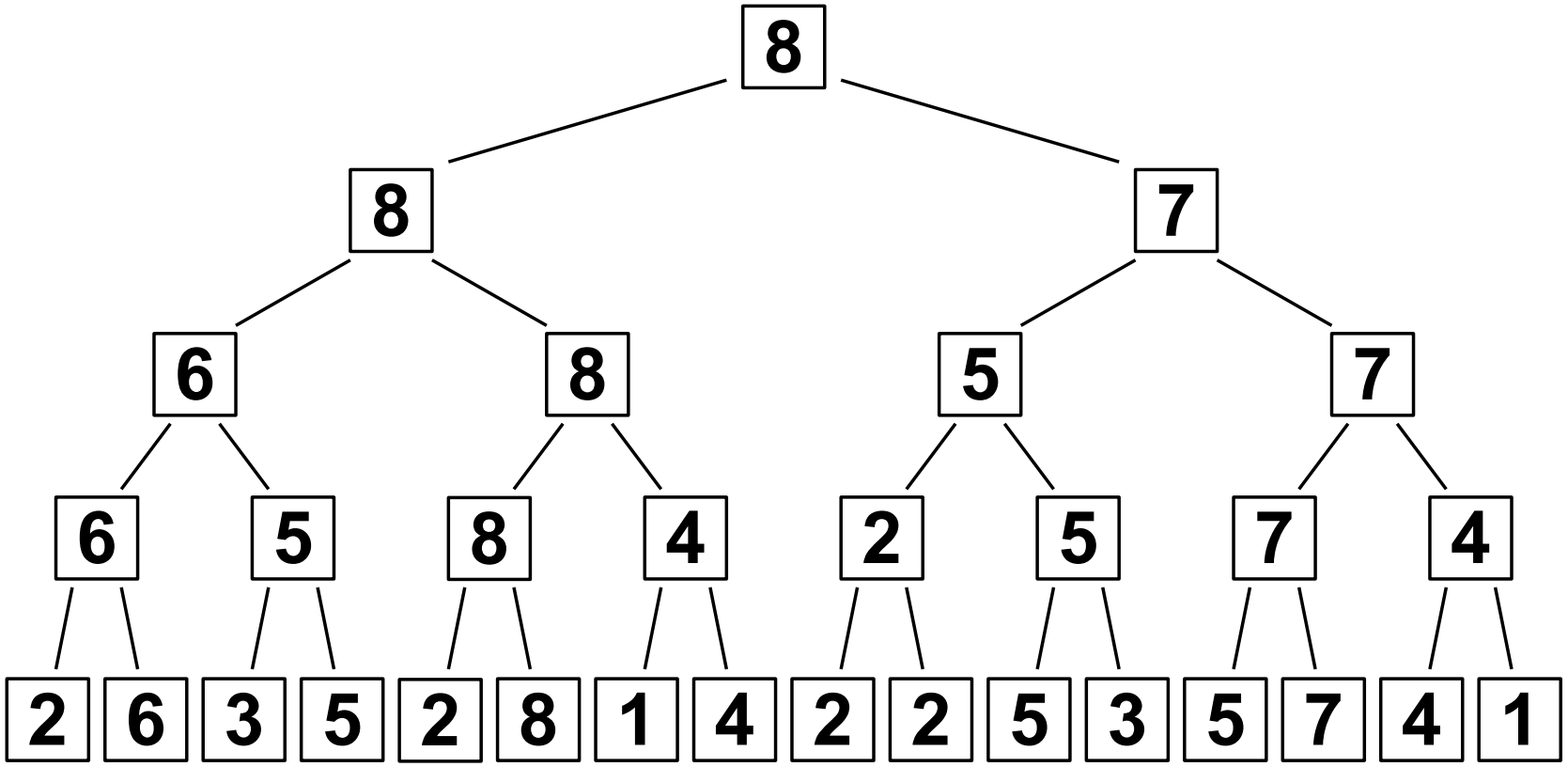
```
maxCachedTree(x: List[[ℤ32]]) =
  if |x| = 0 then NullNode(−∞)
  elif |x| = 1 then SingletonNode(x_0)
  else
    (p, q) = x.split()      ⊛ into approximately equal pieces
    (a, b) = (maxCachedTree p, maxCachedTree q)      ⊛ parallel recursion here
    PairNode(a.val MAX b.val, a, b)
  end
```

Code to Make a MAX-cached Tree (2 of 2)

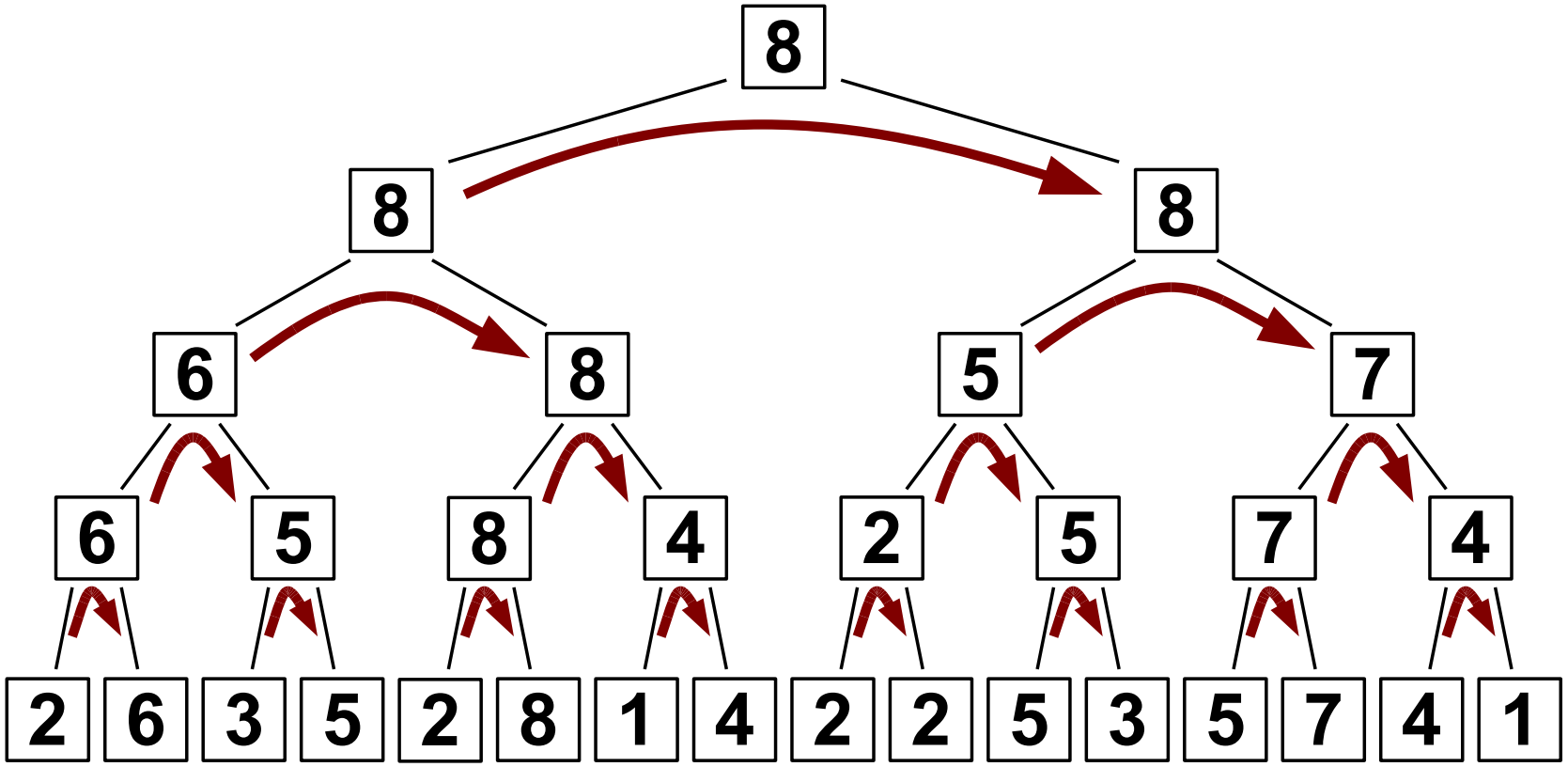
```
trait CachedTree comprises { NullNode, SingletonNode, PairNode }
  getter val(): ℤ32
end

object NullNode(val: ℤ32) extends CachedTree end
object SingletonNode(val: ℤ32) extends CachedTree end
object PairNode(val: ℤ32, a: CachedTree, b: CachedTree) extends CachedTree end
```

```
maxCachedTree(x: List[[ℤ32]]) =
  if |x| = 0 then NullNode(−∞)
  elif |x| = 1 then SingletonNode(x_0)
  else
    (p, q) = x.split()      ⊛ into approximately equal pieces
    (a, b) = (maxCachedTree p, maxCachedTree q)      ⊛ parallel recursion here
    PairNode(a.val MAX b.val, a, b)
  end
```

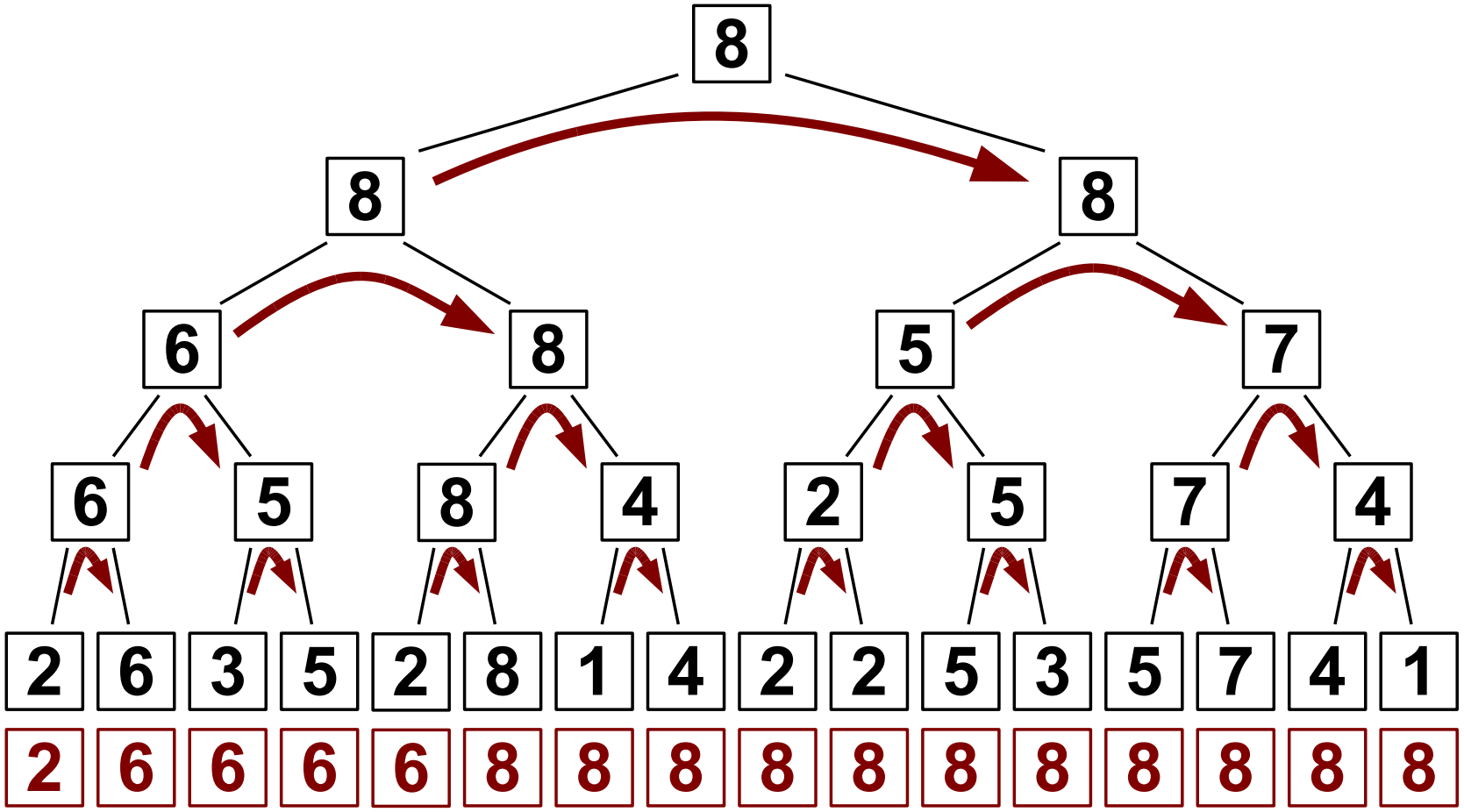
Parallel Left-to-Right Sweep (1 of 3)



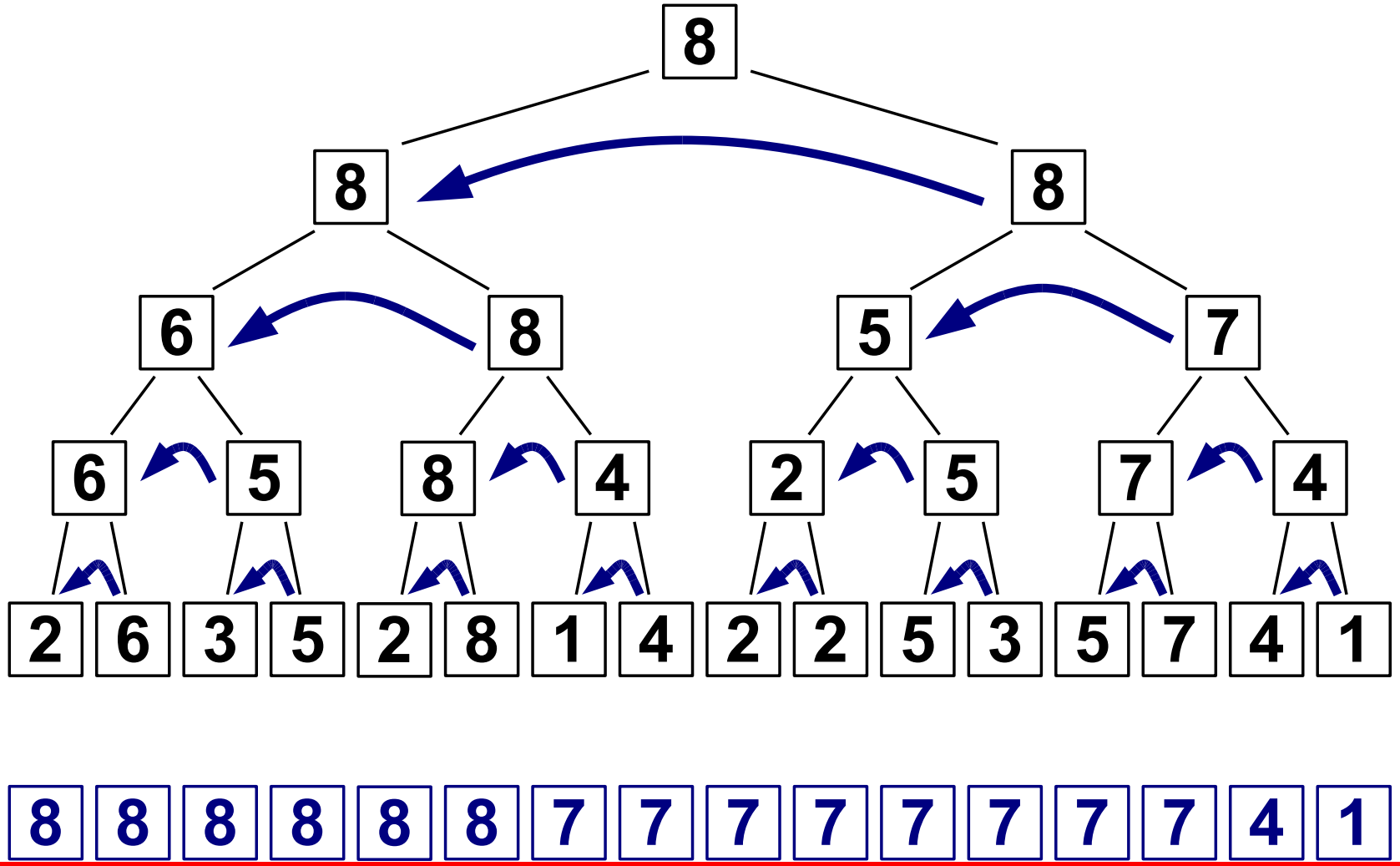
Parallel Left-to-Right Sweep (2 of 3)



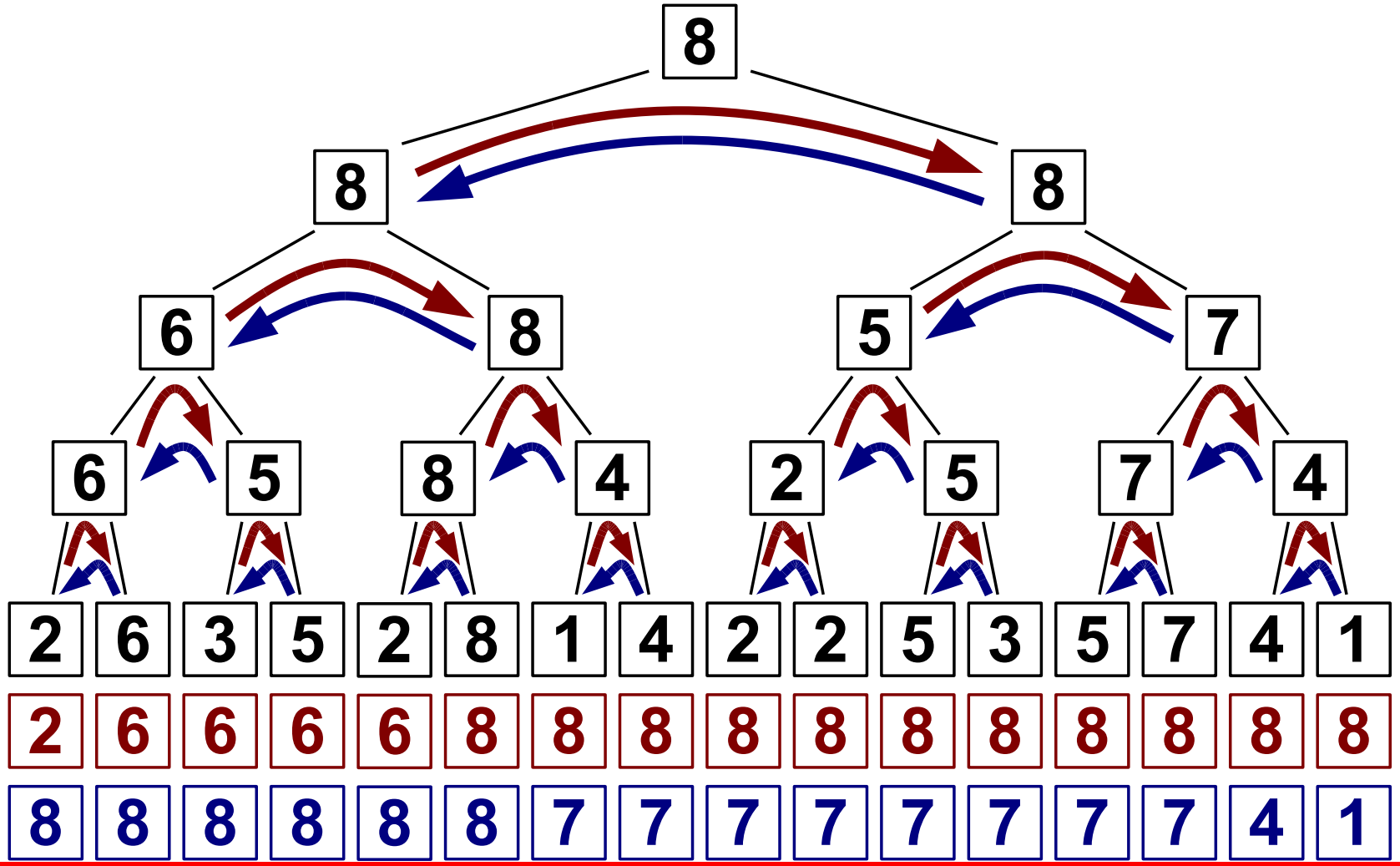
Parallel Left-to-Right Sweep (3 of 3)



Parallel Right-to-Left Sweep



Both Sweeps at Once



Efficient Parallel Solution (Two Passes)

$$histogramWater\big(x\colon \mathrm{List}[\![\mathbb{Z}32]\!]\big)\colon \mathbb{Z}32 = process(maxCachedTree\, x, -\infty, -\infty)$$

$$process(x\colon \mathrm{PairNode}, left\colon \mathbb{Z}32, right\colon \mathbb{Z}32)\colon \mathbb{Z}32 =$$

⊛ parallel recursion here

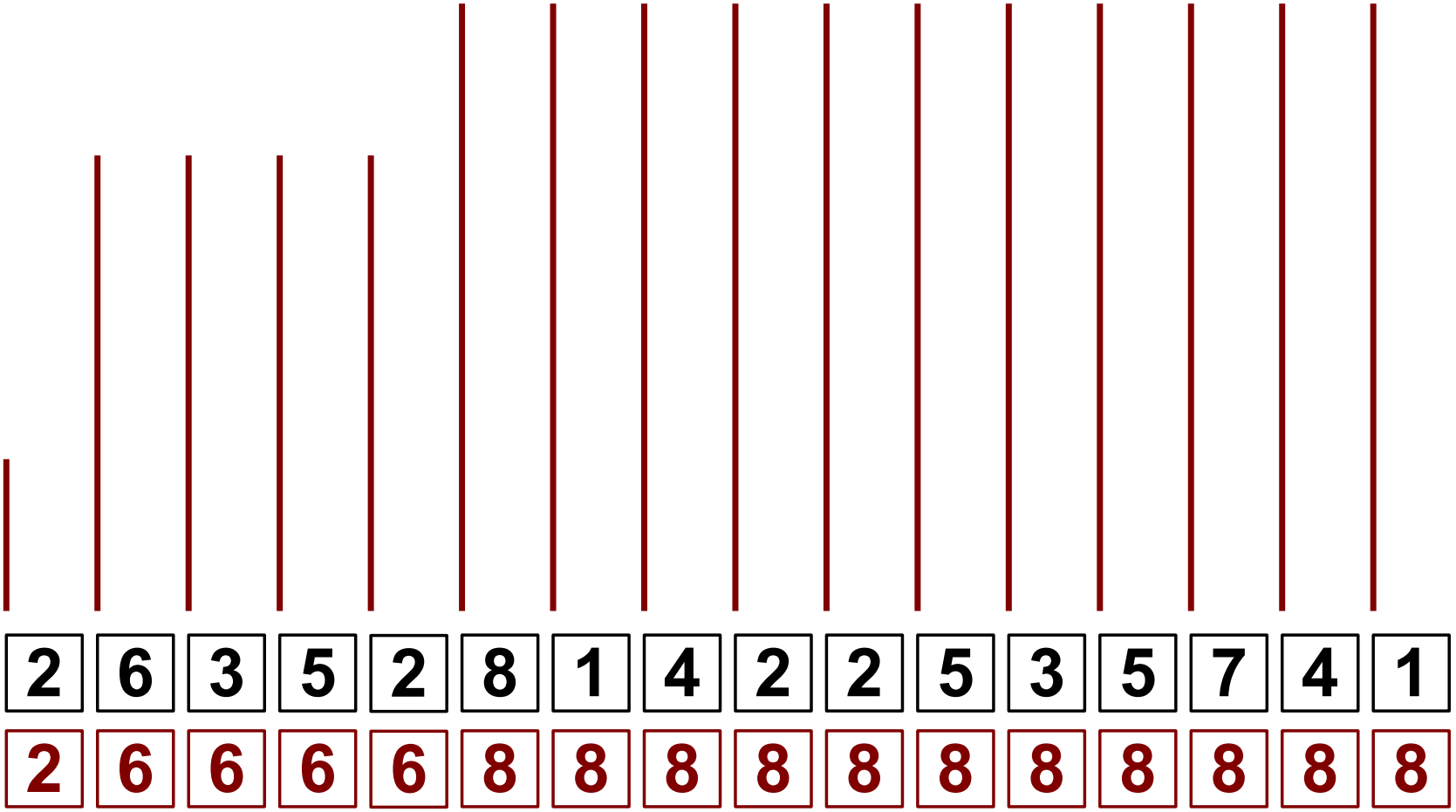
$$process(x.a, left, x.b.val\ \texttt{MAX}\ right) + process(x.b, left\ \texttt{MAX}\ x.a.val, right)$$

$$process(x\colon \mathrm{SingletonNode}, left\colon \mathbb{Z}32, right\colon \mathbb{Z}32)\colon \mathbb{Z}32 =$$

$$\big((left\ \texttt{MIN}\ right)\ \texttt{MAX}\ x.val\big) - x.val$$

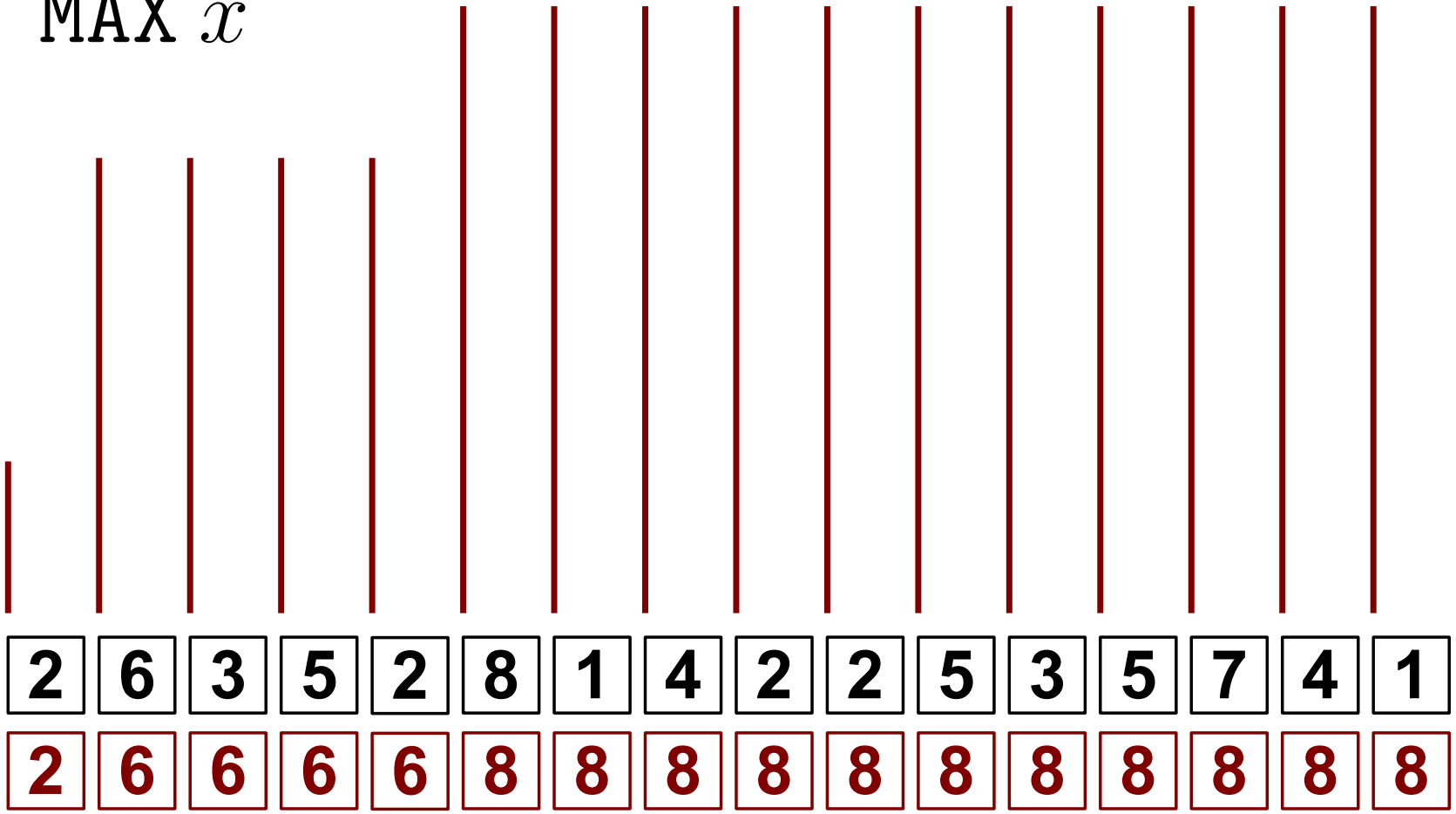
$$process(x\colon \mathrm{NullNode}, left\colon \mathbb{Z}32, right\colon \mathbb{Z}32)\colon \mathbb{Z}32 = 0$$

Left-to-Right Sweep

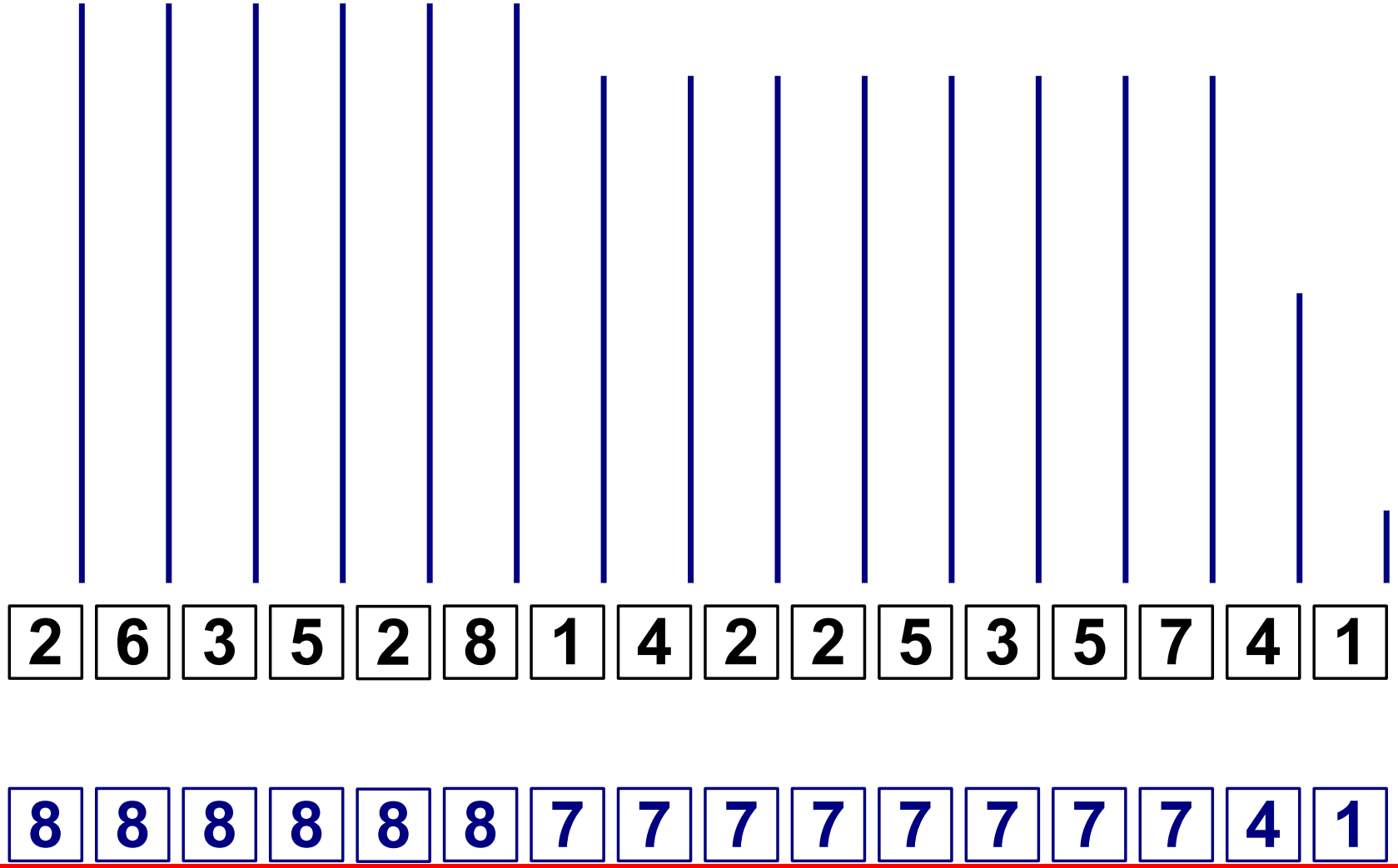


Left-to-Right Sweep: Parallel Prefix

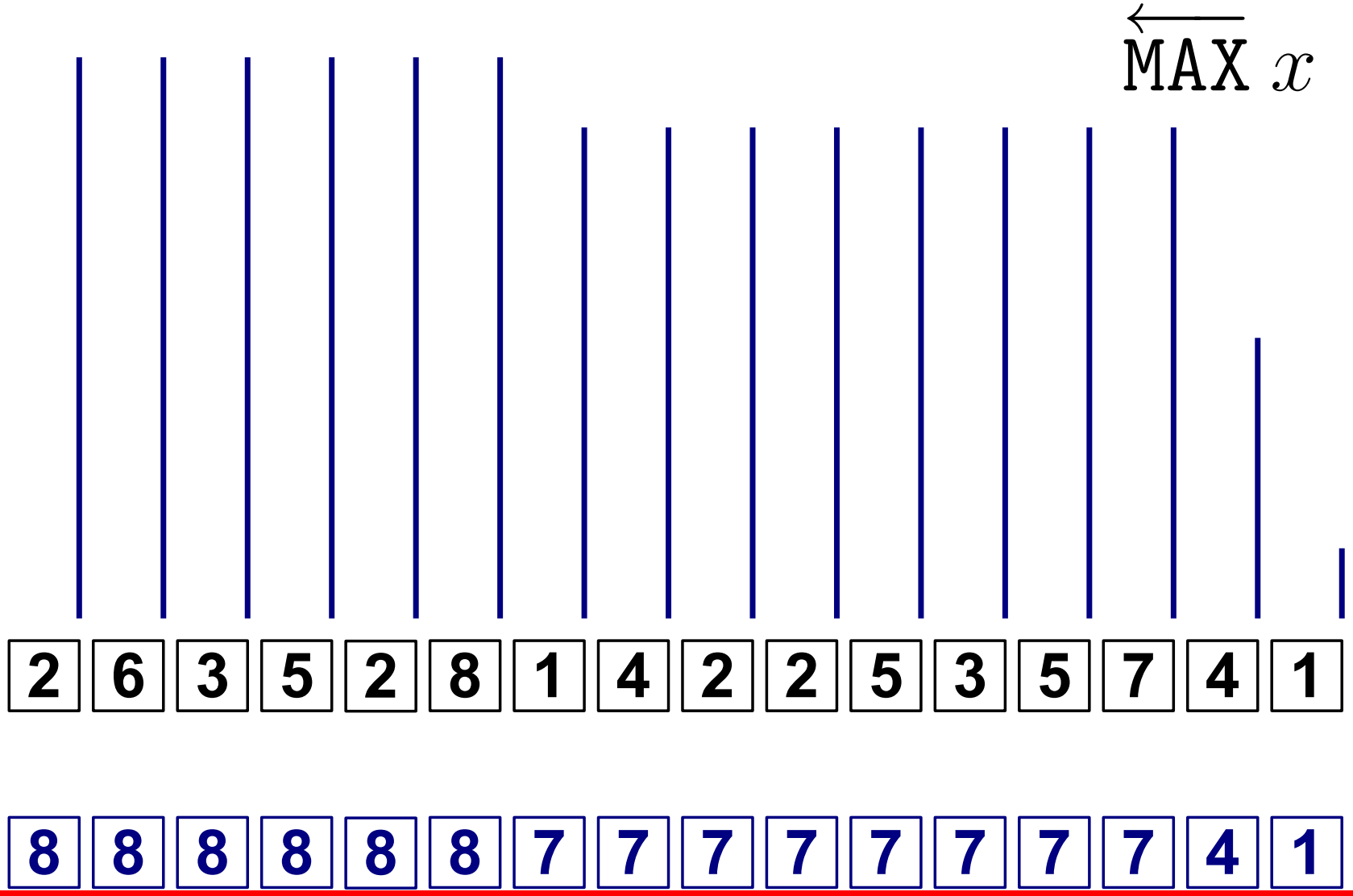
$\overrightarrow{\mathrm{MAX}}\ x$



Right-to-Left Sweep



Right-to-Left Sweep: Parallel Suffix



Concise Solution (1 of 6)

$$histogramWater\big(x\colon \mathrm{List}[\![\mathbb{Z}32]\!]\big)\colon \mathbb{Z}32 = \sum_{(v,left,right)\leftarrow zip(x,\overrightarrow{\texttt{MAX}}\ x,\overleftarrow{\texttt{MAX}}\ x)} \big((left\ \texttt{MIN}\ right) - v\big)$$

Concise Solution (2 of 6)

$$histogramWater\left(x\colon \mathrm{List}[\![\mathbb{Z}32]\!]\right)\colon \mathbb{Z}32 = \sum_{(v,left,right)\leftarrow zip(x,\overrightarrow{\texttt{MAX}}\ x,\overleftarrow{\texttt{MAX}}\ x)} \left((left\ \texttt{MIN}\ right) - v\right)$$

Concise Solution (3 of 6)

$$histogramWater\left(x\colon \mathrm{List}[\![\mathbb{Z}32]\!]\right)\colon \mathbb{Z}32 = \sum_{(v,left,right)\leftarrow zip(x,\overrightarrow{\texttt{MAX}}\ x,\overleftarrow{\texttt{MAX}}\ x)} \left((left\ \texttt{MIN}\ right) - v\right)$$

Concise Solution (4 of 6)

$$histogramWater\left(x\colon \mathrm{List}[\![\mathbb{Z}32]\!]\right)\colon \mathbb{Z}32 = \sum_{(v,left,right)\leftarrow zip(x,\overrightarrow{\texttt{MAX}}\ x,\overleftarrow{\texttt{MAX}}\ x)} \left((left\ \texttt{MIN}\ right) - v\right)$$

Concise Solution (5 of 6)

$$histogramWater\left(x\colon \mathrm{List}[\![\mathbb{Z}32]\!]\right)\colon \mathbb{Z}32 = \sum_{(v,left,right)\leftarrow zip(x,\overrightarrow{\texttt{MAX}}\ x,\overleftarrow{\texttt{MAX}}\ x)} \left((left\ \texttt{MIN}\ right) - v\right)$$

Inlining and loop fusion transform this into the efficient two-pass sequential solution.

Concise Solution (6 of 6)

$$histogramWater\big(x\colon \mathrm{List}[\![\mathbb{Z}32]\!]\big)\colon \mathbb{Z}32 = \sum_{(v,left,right)\leftarrow zip(x,\overrightarrow{\texttt{MAX}}\ x,\overleftarrow{\texttt{MAX}}\ x)} \big((left\ \texttt{MIN}\ right) - v\big)$$

Inlining and loop fusion transform this into the efficient two-pass sequential solution.

Inlining and deforestation (“loop fusion on trees”) transform this into the efficient two-pass parallel solution.

Algebraic Properties Are Important!

- Associative
- Commutative
- Idempotent
- Identity
- Zero

Algebraic Properties Are Important!

- Associative: grouping doesn't matter!
- Commutative: order doesn't matter!
- Idempotent: duplicates don't matter!
- Identity: this value doesn't matter!
- Zero: other values don't matter!

Invariants give the implementation *wiggle room*, that is, the freedom to exploit alternate representations and implmentations.

In particular, *associativity* gives implementations the necessary wiggle room to use parallelism—*or not*—as resources dictate.

The Big Idea

- Loops and summations and list/set comprehensions are alike!

$$\texttt{for } i \leftarrow 1:1000000 \texttt{ do } x_{\mathrm{i}} := x_{\mathrm{i}}^2 \texttt{ end}$$

$$\sum_{i \leftarrow 1:1000000} x_{\mathrm{i}}^2$$

$$\langle\, x_{\mathrm{i}}^2 \mid i \leftarrow 1:1000000 \,\rangle, \{\, x_{\mathrm{i}}^2 \mid i \leftarrow 1:1000000 \,\}$$

  - Generate an abstract collection
  - The *body* computes a function of each item (`map`)
  - Combine the results (or just synchronize) (`reduce`)
- Whether to be sequential or parallel is a separable question
  - That's why they are especially good abstractions!
  - Make the decision on the fly, to use available resources

Parallelism Is Like Memory Management

- Resource management problems throughout history:
  - Registers: register allocators
  - Main memory: overlays, virtual memory, GC heaps
  - Cache: cache-oblivious algorithms, self-tuning algorithms
- The key is to maintain an invariant that gives the implementation some wiggle room!
- A good programming language or environment aids or enforces those invariants.
- We need to do for processor allocation what garbage collection has done for memory allocation.

Conclusion

- A program organized according to linear problem decomposition principles can be really hard to parallelize.
- A program organized according to independence and divide-and-conquer principles is easily run either in parallel or sequentially, according to available resources.
- The new strategy has costs and overheads. They will be reduced over time but will not disappear.
- In a world of parallel computers of wildly varying sizes, this is our only hope for program portability in the future.
- Better language design can encourage "independent thinking" that allows parallel computers to run programs effectively.


ORACLE®