Efficient Algorithms on Sparse Numbers

Jean Vuillemin

École Normale Supérieure, Paris

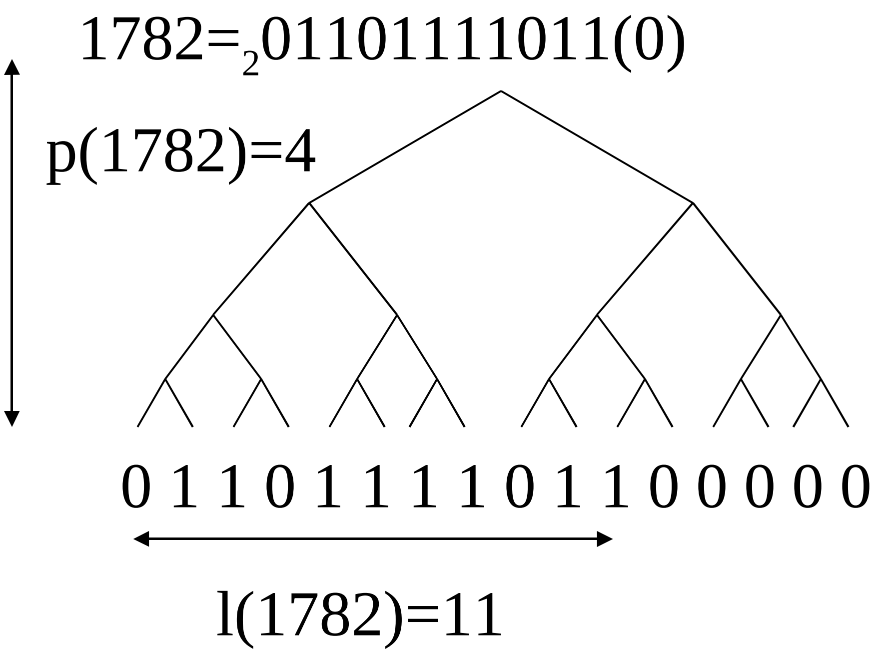
- **Integer as Binary Sequence, Tree, DAG.**
- **Two digits arithmetic.**
- **Dense Compare/Increment.**
- **Dense vs. Sparse Number.**
- **Sparse memo Sum/Product/Others.**
- **Demo/Conclusions.**

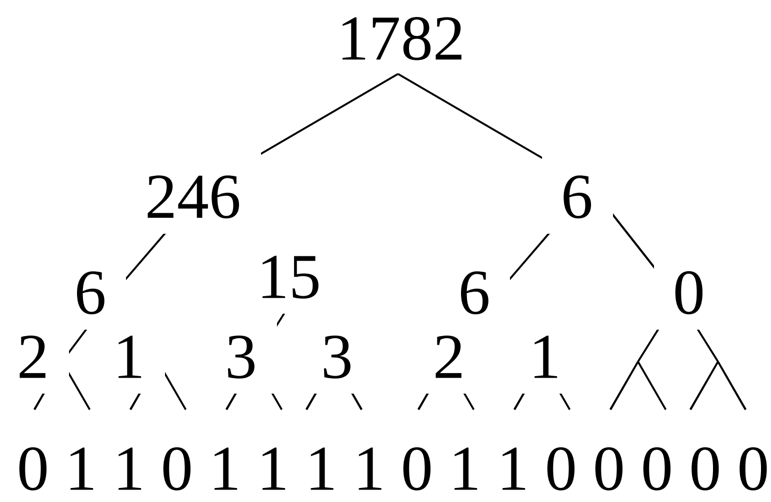
Binary Integers as Binary Trees

Binary decomposition $n = n_0 + 2n_1$ has length $l(n) = \lceil \log_2(n+1) \rceil$.

Dichotomy decomposition $n = n_0 + n_1 \beta_p$ in base $\beta_p = 2^{2^p}$

has depth $p(n) = l(l(n)+1) = \lfloor \log_2 \log_2 (n+1) \rfloor$





Two Digits Arithmetic

$$mul(0)(a,b,c,d) = a\ ?\ b + c + d$$

$$mul(n+1)(a,b,c,d) = q_0 + \beta_n s_0 + \beta_{n+1} p \ \{$$

$$q = mul(n)(a_0, b_0, c_0, d_0)$$

$$r = mul(n)(a_0, b_1, c_1, q_1)$$

$$s = mul(n)(a_1, b_0, c_1, q_1)$$

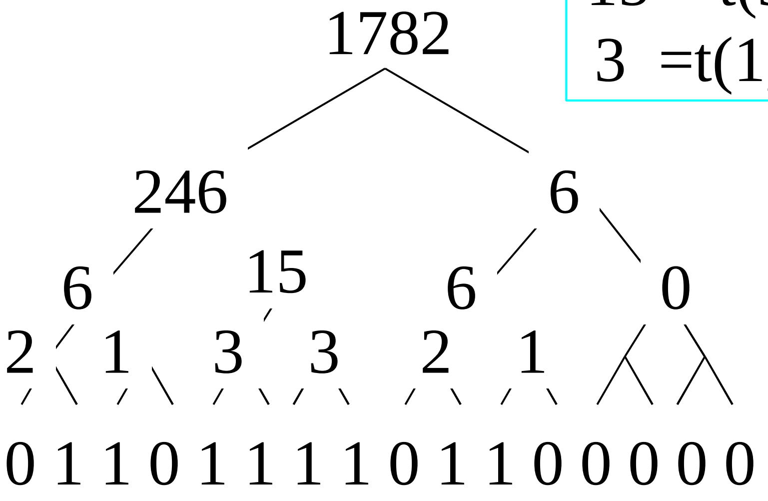
$$p = mul(n)(a_1, b_1, s_1, r_1) \ \}$$

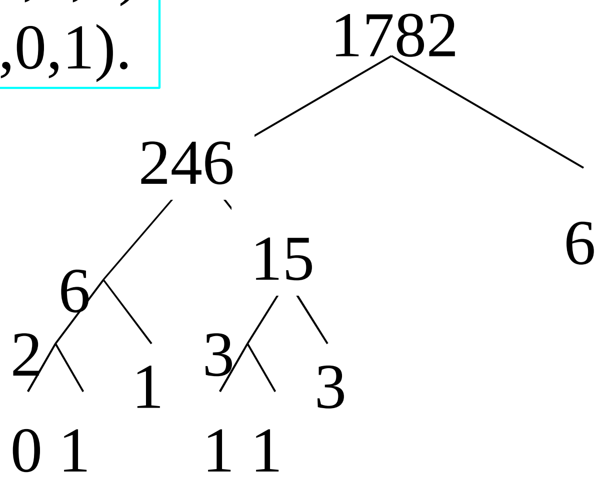
Arithmetic operations can all be lifted from Binary to Tree representation: same complexity.

Trichotomy

1. Recursively decompose $\boldsymbol{n = n_0 + n_1 \beta_p}$
1. Share equal nodes

1782 = t(246,3,6),
246 = t(6,2,15),
15 = t(3,1,3), 6=t(2,1,1)
3 =t(1,0,1), 2=t(0,0,1).





Mersenne/Fermat nodes

$$n = n_0 + n_1 \beta_p$$

$$\beta_p = 2^{2^p}$$

$$M_p = 2^{2^p} - 1$$

$$F_p = 2^{2^p} + 1$$

4 node types:

$n = 0$? $\quad n = zero;$

$n = M_p$? $\quad n = m(p);$

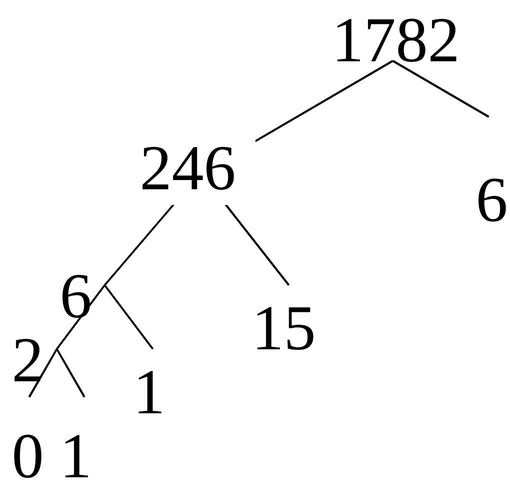
$n_0 = n_1$? $\quad n = f(n_0, p);$

n_0 ? n_1 ? $\quad n = t(n_0, p, n_1).$

22b file!

n3 = m(n1)
n2 = f(1, n1)
n1 = t(0,n0,1)
n0 = m(1)

1782 = t(246,3,6)
246 = t(6,2,15)
15 = m(2)
6=t(2,1,1)
3=m(1)
2=t(0,0,1)



$M_{256} = m(\beta_3)$
$F_{256} = f(1, \beta_3)$
$\beta_3 = t(0,3,1)$
3= m(1)

Size of Numbers

0=zero	0
1=m(0)	1
2=t(0,0,1)	2
3=m(1)	2
4=t(0,1,1)	2
5=f(1,1)	2
6=t(2,1,1)	3
7=t(3,1,1)	3
8=t(0,1,2)	3
9=t(1,1,2)	3

$$s(n) \; ? \; l(n)$$

$$l(n) = ?\log_2(n+1)?$$

$$s(1 \quad n) = n$$

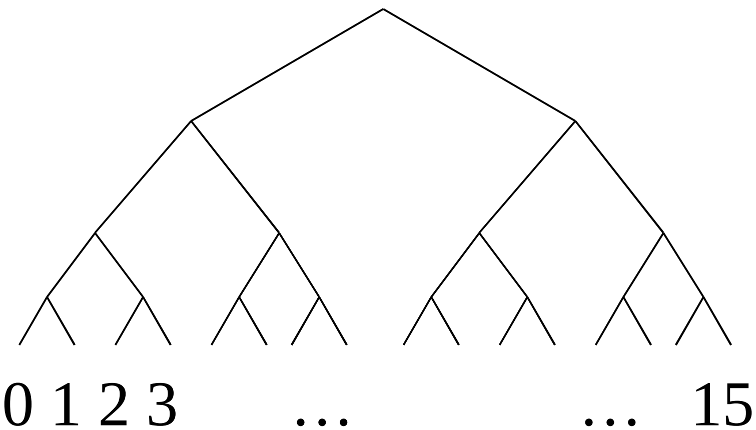
10=f(2,1)	3
11=t(3,1,2)	4
12=t(0,1,3)	3
13=t(1,1,3)	3
14=t(2,1,3)	4
15=m(2)	3
16=t(0,2,1)	3
17=f(1,2)	3
18=t(2,2,1)	3
19=t(3,2,1)	4

Sparse & Dense Numbers

$s(n) \; ; \; w = k_p \frac{l}{p-c}$

Worst: $1 \, ? \, k_p \, ? \, 2$

$p = c + 2^c$



$s(n) = 30$

$l(n) = 64$

$p(n) = 6$

$c(n) = 2$

Average: $s(n) \; ; \; r_p w$

$1 - \frac{1}{2e} \, ? \, r_p \, ? \, 1$

Sparse: $s(n) \quad l(n)$

n is sparse ⇘
bin(***n***) has low entropy.

Primitive Operations

Depth $n_p = p(n) - 1$

Division by $\beta_p = 2^{2^p}$

Equality test $n = m \Leftrightarrow n_h = m_h$

all in 1 machine operation.

$$d = 0 \mid cons(g, p, 0) \leftarrow g;$$

Constructor $g + d\beta_p$

$$g_p \mid p \leftarrow cons(g_0, p, add(d, g_1)),\ (g_0, g_1) \leftarrow div(g, 2^{2^p});$$

$$d_p \mid p \leftarrow cons(l, add(p, 1), d_1),\ l \leftarrow cons(g, p, d_0),\ (d_0, d_1) = div(d, 2^{2^p});$$

$$g = d = M_{p-1} \mid m(p)$$

$$g = d \mid f(g, p)$$

Node constructors: if new!

$$g \mid d \neq t(g, p, d)$$

Fast Dense Operations

In $p(n)$ operations.

$$bit(2^k, n) = (k = n_p)\ ?\ bit(0, n_1)\ :$$
$$(k < n_p)\ ?\ bit(2^k, n_0) : 0$$

$$bit(0,0) = 0$$
$$bit(0, m(n)) = 1$$
$$bit(0, t(l, p,)) = bit(0, l)$$

$$comp(n,m) = (n = m)\ ?\ 0 :$$
$$(n_p\ ?\ m_p)\ ?\ comp(n_p, m_p) :$$
$$(n_1\ ?\ m_1)\ ?\ comp(n_1, m_1) : comp(n_0, m_0)$$

Comparison is exponentially faster than classical, in the worst case.

$$\nu(0) = 0,\ \nu(1) = 1$$

Number of 1 in binary representation.

$$\nu(m(p)) = t(0, p, 1)$$
$$\nu(f(g, p)) = 2\nu(g)$$
$$\nu(t(g, p, d)) = \nu(g) + \nu(d)$$

Linear time through memo function.

Sparse Increment

$add(0,n) =$ n

$add(1,0) =$ 1

$add(1,1) =$ 2=t(0,0,1)

$add(1,t(g,p,d)) = cons(add(1,g), p, d)$

Increment $n=M_p$ in 2^p operations.

$add(1,m(n)) =$ t(0,n,1)

Increment in p operations.

$add(1,t(m(p-1),p,r)) = t(0,p,add(1,r))$

$add(1,t(l,p,r)) = t(add(1,l), p, r)$

It implies that $s(n,n+1)<s(n)+p(n)$.

Some neighborhood of a sparse number is sparse.

Sparse Twice

$$add(n,n) = mul(2,n)$$
$$mul(2,0) = 0$$
$$mul(2,1) = 2$$
$$mul(2,t(l,p,r)) = cons(mul(2,l), p, mul(2,r))$$

This version of doubling may require up to *l(n)* operations.

For sparse Trees, time can get exponential in size!

Implement ***mul*** as a memo function: computed values are always stored. They get retrieved whenever possible; otherwise, the computation is performed ***exactly once***.

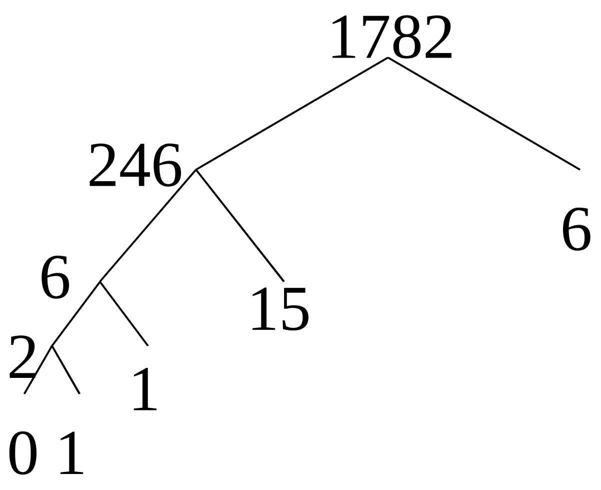
Time & space for twice ***mul(2,n)*** with memo is linear in ***s(2n)***. Hence ***s(2n)<2s(n)***.

Over 10^6 numbers in the expression-neighborhood of F_{1024} are very sparse: ***s(n)<p(n)***.

Sparse Product

Time for memo sparse product *nm* is at most *s(n)s(m).*

Sometimes less:



$$1782^2 = 246^2 + 2\beta_3(6x246) + \beta_4 6^2$$

$$246^2 = 6^2 + 2\beta_2(6x15) + \beta_3 15^2$$

$$6x246 = 6^2 + \beta_2(6x15)$$

$$15^2 = \beta_3 - 2\beta_2 + 1$$

$$6x15 = 6\beta_2 - 6$$

$$6^2 = 4 + 2\beta_2$$

Dictionary

1. Code set $s=\{n_1 \ldots n_k\}$ by number $N_s=\Sigma\, 2^{n_k}$.
2. Map dictionary operations to Boolean algebra over integers.

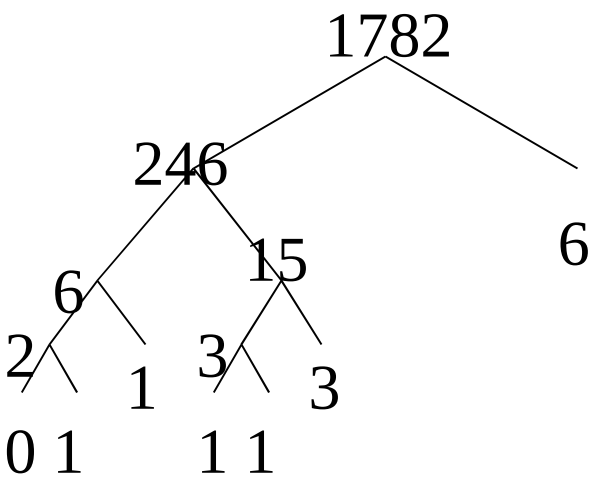
Supports ***Search/Insert/Delete/Min/Max/Complement***
in $p(N_s)<l(k)+p(n_k)$ operations.

Supports ***Merge/Intersect/XOR/Shift &*** more.
- Linear time/space with respect to un-shared size for dense numbers.
- Quadratic time/space with respect to shared size for sparse numbers.

Can be used to efficiently implement the hash-table required to maintain unique numbers n_h.

Related to BDD & BMD

Boolean operations on sparse numbers are closely related to those on the Binary Moment Diagram BMD, and to those on the Binary Decision Diagram BDD of Randy Bryant.



$f_{1782}(x_1x_2x_3x_4)=f_6+x_3\, f_{15}+x_4\, f_6$

$f_{15}=(1-x_1)(1-x_2)$

$f_6=x_1+x_2$

Here + and - mean XOR $\oplus$.

Conclusions

Demo?

1. Time/space with two-digit arithmetic on Trees is less than a constant *c* off from Sequences.

2. Terminating the recursion at the right Tree depth guaranties a small *c*.

**3. DAG Space is smaller than that of Sequence.
DAG Time is smaller than that of Tree.**

4. DAG can sex-up a BigNum packages, at small cost in programming/efficiency. Throw in Dictionary/Hash-Table/BDD as well!

5. BDD & BigNum package performance hinges on efficient storage allocation: combine DAGs with buddy system memory.