Iterator

The Iterator package provides Maple objects that implement fast methods for iterating over discrete structures.

```
> with(Iterator)
```

[*BinaryGrayCode*, *BinaryTrees*, *BoundedComposition*, *CartesianProduct*, *Chase*, *Combination*, *Count*, *FillRucksack*, *Inversion*, *MixedRadix*, *MixedRadixGrayCode*, *MixedRadixTuples*, *MultiPartition*, *NearPerfectParentheses*, *NestedParentheses*, *OrientedForests*, *Partition*, *PartitionFixedSize*, *Permute*, *Product*, *Reverse*, *RevolvingDoorCombination*, *Select*, *SetPartitionFixedSize*, *SetPartitions*, *SplitRanks*, *TopologicalSorts*, *Trees*]

Integer Divisors

An efficient way to compute all divisors, given a prime factorization $b_1^{e_1} \cdot b_2^{e_2} \cdot \ldots \cdot b_m^{e_m}$, is to loop through the exponents with a mixed-radix Gray code with radices $r_j = e_j + 1$, and then multiply or divide the previously computed divisor by b_j, depending on whether the j-th exponent is increased or decreased.

```
> p := 12*5*8*24*13*9;
```

$$p := 1347840$$

```
> f := ifactors(p)[2];
```

$$f := [[2, 8], [3, 4], [5, 1], [13, 1]]$$

Extract the lists of the bases and exponents.

```
> b := map2(op,1,f);
  e := map2(op,2,f);
```

$$b := [2, 3, 5, 13]$$
$$e := [8, 4, 1, 1]$$

Create a MixedRadixGrayCode iterator using the append_change option so the last index contains a signed value of the index that changed (the initial value of the index is 0).

```
> G := MixedRadixGrayCode(e +~ 1, append_change):
```

Get the position of the index that indicates the change. The output method of the iterator object, G, returns the Array that is used to output the results.

```
> n := upperbound(output(G)):
```

Update and return the divisor (d) given g, a vector whose n-th slot stores the index of the prime that changed, with a positive value indicating an increase by one and a negative value, a decrease by one.

```
> update_d := proc(g,b,n)
```

```
global d;
local i;
    i := g[n];
    if   i < 0 then d := d/b[-i];
    elif i > 0 then d := d*b[+i];
    else            d := 1;
    end if;
    return d;
end proc:
```

Use the iterator and procedure to compute all divisors.

```
> [seq(update_d(g,b,n), g = G)];
```

[1, 2, 4, 8, 16, 32, 64, 128, 256, 768, 384, 192, 96, 48, 24, 12, 6, 3, 9, 18, 36, 72, 144, 288, 576, 1152, 2304, 6912, 3456, 1728, 864, 432, 216, 108, 54, 27, 81, 162, 324, 648, 1296, 2592, 5184, 10368, 20736, 103680, 51840, 25920, 12960, 6480, 3240, 1620, 810, 405, 135, 270, 540, 1080, 2160, 4320, 8640, 17280, 34560, 11520, 5760, 2880, 1440, 720, 360, 180, 90, 45, 15, 30, 60, 120, 240, 480, 960, 1920, 3840, 1280, 640, 320, 160, 80, 40, 20, 10, 5, 65, 130, 260, 520, 1040, 2080, 4160, 8320, 16640, 49920, 24960, 12480, 6240, 3120, 1560, 780, 390, 195, 585, 1170, 2340, 4680, 9360, 18720, 37440, 74880, 149760, 449280, 224640, 112320, 56160, 28080, 14040, 7020, 3510, 1755, 5265, 10530, 21060, 42120, 84240, 168480, 336960, 673920, 1347840, 269568, 134784, 67392, 33696, 16848, 8424, 4212, 2106, 1053, 351, 702, 1404, 2808, 5616, 11232, 22464, 44928, 89856, 29952, 14976, 7488, 3744, 1872, 936, 468, 234, 117, 39, 78, 156, 312, 624, 1248, 2496, 4992, 9984, 3328, 1664, 832, 416, 208, 104, 52, 26, 13]

Octacode

The octacode, O_8, a linear self-dual code of length 8 over $\mathbb{Z}_4$ and minimal Lee distance 6, can be computed from the generator polynomial

> $g := x^3 + 2x^2 + x - 1$:

as $O_8 = \sum_{u = g \cdot v} \prod_{k=0}^{7} z_{u_k}$ with $v = \sum_{k=0}^{3} v_k \cdot x^k$, $u = \sum_{k=0}^{6} u_k \cdot x^k$, $g \cdot v \ \mathbf{mod}\ 4 = u$, with u_7 selected so $\sum_{k=0}^{7} u_k = 0 \ \mathbf{mod}\ 4$ and $0 \le v_0, v_1, v_2, v_3 < 4$.

Assign a procedure that computes one term of the sum, corresponding to the vector *v* = $[v_0, v_1, v_2, v_3]$.

```
> pterm := proc(v)
  local u,u7;
  global g, x, z;
      u := [coeffs(collect(g*add(v[k+1]*x^k, k=0..3),x),x)];
      u7 := modp(-add(u),4);
      mul(z[modp(u,4)],u=u) * z[0]^(7-numelems(u)) * z[u7];
   end proc:
```

Use the MixedRadixTuples iterator and sum over all values of *v*.

```
> V := Iterator:-MixedRadixTuples([4,4,4,4]):
> O__8 := add(pterm(v), v = V);
```

$$O_8 := z_0^8 + 14\, z_0^4 z_2^4 + 56\, z_0^3 z_1^3 z_2 z_3 + 56\, z_0^3 z_1 z_2 z_3^3 + 56\, z_0 z_1^3 z_2^3 z_3 + 56\, z_0 z_1 z_2^3 z_3^3 + z_1^8 + 14\, z_1^4 z_3^4 + z_2^8 + z_3^8$$