Estimate function for compression

```
PROCEDURE Estimate(c) {
    IF c.o.v.m > 0  && c.p.v.m > 0 DO{
        pred = (c.n.v.d+c.p.v.d-c.o.v.d); delta = c.v.g – pred;}        # a, b and d are known (case 1)
    ELSE  IF c.o.v.m > 0  DO{
        pred = (2*c.n.v.d -c.o.v.d); delta = c.v.g – pred;}             # a and d are known (case 2)
    ELSE  IF c.n.v.m > 0  && c.p.v.m > 0  DO{
        pred = (c.n.v.d +c.p.v.d)/2; delta = c.v.g – pred;}             # a and b are known (case 3)
    ELSE  IF c.n.v.m > 0  DO{
        pred = c.n.v.d ; delta = c.v.g – pred;}                         # a is known (case 4)
    ELSE  IF c.p.v.m > 0  DO{
        pred = c.p.v.d;  delta = c.v.g – pred;}                         # b is known (case 5)
    ELSE{
        pred =  {0,0,0}; delta =  c.v.g - pred;}                        # nothing is known (case 6)
    D[c.v] = delta + pred;                                              # remember  recomputed vertex for next predictions
    WRITE(vertices, delta);}                                            # store estimate in the vertices file
```

Estimate function for decompression

```
PROCEDURE VERTEX  Estimate(c) {
    delta = READ(vertices);                                                          # read next vertex delta
    IF c.o.v.m > 0  && c.p.v.m > 0 DO RETURN  delta +  (c.n.v.g+c.p.v.g-c.o.v.g);    # a, b and d are known (case 1)
    IF c.o.v.m > 0  DO RETURN   delta + (2*c.n.v.g -c.o.v.g);                        # a and d are known (case 2)
    IF c.n.v.m > 0  && c.p.v.m >0  DO RETURN  delta + (c.n.v.g +c.p.v.g)/2;          # a and b are known (case 3)
    IF c.n.v.m > 0  DO RETURN  delta + c.n.v.g;                                      # a is known (case 4)
    IF c.p.v.m > 0  DO RETURN  delta + c.p.v.g;                                      # b is known (case 5)
    RETURN delta;   }                                                                # nothing is known (case 6)
```