HAMMING CODES

Αν ένας αριθμός με B bits είναι μικρότερος γραμμένος στη δύναμη 2^B θα είναι και μικρότερος γραμμένος στη δύναμη 10(αν κάνεις fill με trailing μηδενικά για να έχουν το ίδιο μέγεθος).
Η άσκηση είναι απλή για να καταλάβουμε τη χρήση των bitwise operators.
Ξεκινάμε από τον αριθμό 0 και βρίσκουμε τον πρώτο αριθμό X που έχει Hamming distance >=D από αυτόν. Αποθηκεύουμε τον αριθμό αυτό σε ένα array. Μετά συνεχίζουμε από το X+1 και γυρεύουμε τον πρώτο αριθμό Y που έχει Hamming distance >=D από όλους τους αριθμούς του array. Επαναλαμβάνουμε τη διαδικασία ξεκινώντας απο Y+1 μέχρι να βρούμε N αριθμούς.
Πως βρίσκουμε το Hamming distance 2 αριθμών. Το Νοστο bit ενός αριθμού X είναι το
Nosto=(1<<(N-1))&X;
Άρα συγκρίνουμε bit by bit.
Ένας άλλος τρόπος είναι να χρησιμοποιήσουμε το XOR(^). Το XOR είναι ένας λογικός τελεστής που δίνει true αν 1 και μόνο 1 απο τα input έιναι true.
XOR
1,1 → 0
1,0 → 1
0,1 → 1
0,0 → 0
Άρα βασικά το XOR επιστρέφει 1(true) μόνο στα διαφορετικά bits. Άρα ο αριθμός των 1 στο a^b είναι το hamming distance τους. Υπάρχουν πολλοί τρόποι να μετρήσουμε τα 1 όπως το built in
int __builtin_popcount (unsigned int x);
ένας άλλος τρόπος να τα μετρήσουμε είναι: το x&(-x) επιστρέφει το least significant bit του x.(δηλαδή το πιο δεξί set(1/true) bit, αν έχουμε 00101010 θα επιστρέψει 00000010)
όποτε κάνοντας x^=(x & (-x)); μέχρι το x να γίνει 0, μετρά πόσα set bits(1) έχει το x.

```
int number_of_ones(int x){
   int ones=0;
   while(x){
      x^=(x & (-x));
      ones++;
   }
   return ones;
}
```

άλλος τρόπος: http://articles.leetcode.com/2010/09/number-of-1-bits.html