Through all this mask subpatterns

Enumerating subpatterns fixed mask

Dana bitmask m. Requires effectively enumerate all its subpatterns, ie such masks s, which can be included only those bits that are included in the mask m.

Immediately look at the implementation of this algorithm, based on tricks with Boolean operations:

```
int s = m;
while (s > 0) {
      ... You can use the s ...
      s = (s-1) & m;
}
```

or by using a more compact operator for:

```
for (int s=m; s; s=(s-1)&m)
      ... You can use the s ...
```

The only exception for both versions of the code - the subpattern is zero, will not be processed. Processing it will take out of the loop, or use a less elegant design, for example:

```
for (int s=m; ; s=(s-1)&m) {
      ...You can use the s ...
      if (s==0)  break;
}
```

Let us examine why the above code really finds all of this mask subpattern, without repetitions in O (number), and in descending order.

Let us have a current capturing subpattern s, and we want to move to the next subpattern. Subtract from the mask $s$unit, thus we remove the rightmost single bit, and all the bits to put it in the right 1. Next, remove all the "extra" bits set, which are not included in the mask m, and therefore can not be included in the subpattern. Removal operation is performed bit $\& m$. As a result, we "cut off the" mask $s-1$before the largest value that it can take, ie until after the next subpattern $s$in descending order.

Thus, the algorithm generates all subpatterns this mask in order strictly decreasing, spending on each transition two elementary operations.

Especially, consider the moment when $s = 0$. After running $s - 1$we get a mask in which all bits are included (bit representation of the number -1), and after removing the extra bits operation $(s-1)\&m$will nothing but a mask m. Therefore, the mask $s = 0$should be careful - if time does not stop at zero mask, the algorithm may enter an infinite loop.

Through all the masks with their subpatterns. Rating3^n

In many problems, especially in the dynamic programming by masks is required to sort out all the masks, and masks for each - all subpatterns:

```
for (int m=0; m<(1<<n); ++m)
	for (int s=m; s; s=(s-1)&m)
		... Use s and m ...
```

We prove that the inner loop will execute total $O(3^n)$iterations.

Proof: 1 way . Consider $i$the first bit. For it, in general, there are exactly three options: it is not included in the mask m(and hence in the subpattern s) it enters m, but is not included s, it is included $m$in the s. Total bits n, so all will be different combinations 3^n, as required.

Proof: 2 way . Note that if the mask $m$has $k$included bits, it will have 2^ksubpatterns. Since the length of the masks $n$with $k$bits have included C_n^k(see "binomial coefficients"), then all combinations will be:

$$\sum_{k=0}^{n} C_n^k 2^k.$$

Calculate this amount. For this we note that it is nothing like the binomial theorem expansion in the expression $(1+2)^n$, ie 3^n, as required.