# Conqueror's batalion

Author(s) of the problem: **unknown**
Contest-related materials by: **Vladimír Koutný**

Let's try to tell somehow how good a position is. If we have two soldiers on the same stair (and no other soldiers), in the next round we will have at most only one soldier but one stair higher. In this way, one soldier on the stair $S$ is equivalent to two soldiers on the stair $S+1$. From now on a soldier on the stair $S$ will have the *value* $2^{N-S}$. The value of a position will be the sum of the values of all the soldiers. Note that all positions, where the Conqueror has won, have the value at least $2^{N-1}$ because there is a soldier on the stair number 1.

### Losing positions

If the value of the position is less than $2^{N-1}$ and the commandant plays optimally, the Conqueror will lose.

Proof: If the value is less then $2^{N-1}$, the Conqueror didn't win yet. Let's take a look at one round of the game. The Conqueror divides his soldiers in some way. The group with the smaller (or equal) value has to have the value less than $2^{N-2}$. The commandant will choose this group to stay and the other to leave. When any soldier moves up one stair, his value doubles. Therefore the value of the new position will be less than $2.2^{N-2} = 2^{N-1}$ and the number of soldiers will decrease. There is a finite number of soldiers, and so the game ends in a finite number of moves. The value of a position will always be less than $2^{N-1}$, also at the end there will be no soldiers and the Conqueror will lose, q.e.d.

### Winning positions

If the value of the position is at least than $2^{N-1}$ and the Conqueror plays optimally, he will win.

Proof: The Conqueror has to divide the soldiers into two groups so that the value of each group will be at least $2^{N-2}$. Then regardless of the commandant's choice the new position will again have the value at least $2^{N-1}$ and there will be less soldiers than before. There is a finite number of soldiers, and so the game will end in a finite number of moves. At the end of the game the value of the position will be at least $2^{N-1}$. This means that some of the soldiers has to stand on the stair 1 and the Conqueror won.

We only need to show, that the Conqueror can always divide the soldiers in the way described above.

Lemma: Let $a_1, \ldots, a_M$ ($2^{N-2} \geq a_1 \geq \ldots \geq a_M$, $M \geq 2$) be powers of two with $\sum_{i=1}^{N} a_i \geq 2^{N-2}$. Then for some $k$ holds $\sum_{i=1}^{k} a_i = 2^{N-2}$.

Proof of the lemma: Induction on $M$. For $M = 2$ the lemma holds. Now let $M > 2$ and $\sum_{i=1}^{M} a_i > 2^{N-2}$. Since the sum and $2^{N-2}$ are multiples of $a_M$, the sum is at least $2^{N-2}+a_M$. This means that $\sum_{i=1}^{M-1} a_i \geq 2^{N-2}$ and we may apply the induction assumption.

From the lemma we get that if the sum of the soldiers' values is at least $2^{N-1}$, we may select the first (e.g. closest to the top of the staircase) few of them so that the sum of their values will be exactly $2^{N-2}$. The Conqueror may also divide his soldiers into these two groups, q.e.d.

The solution is a straightforward implementation of the ideas above. We will have to implement arithmetics with big numbers to count the value of a position. If we are in a losing position, we choose an empty set and loose instantly. If we are in a winning position, we find the place where to split the soldiers by adding the values of soldiers on stairs 1, 2, … until the value reaches $2^{N-2}$. Each operation with the big numbers we need can be implemented to run in $O(N)$ time, also the time required for one move is $O(N^2)$.

We could pre-compute some data at the beginning of the game to have a faster algorithm. Let $s_i$ be the number of soldiers on the stair $i$. We will pre-compute the sums $T_k = \sum_{i=1}^{k} s_i 2^{N-i}$. Let's number the soldiers from 1 to some $M$ ordered by the stair, where they stand at the beginning of the game. If we play the game as described above, we may always split the soldiers so that both groups have a consecutive set of numbers. We will keep track of the current numbers of soldiers on each stair and we will keep two integers pointing to the uppermost and lowermost soldier. In fact, we won't move the soldiers upwards, because we know, that after $r$ rounds they are exactly $r$ stairs higher.

From the partial sums we have we are able to compute the value of any consecutive set of soldiers in $O(N)$ time. Using this information we may binary search the place where to split the soldiers. This solution also requires $O(N^2)$ time for pre-computing the partial sums and then $O(N \log N)$ time for each move.

The library your programs played against used the commandant's strategy described above, with the difference that if both possibilities led to a winning position for your program, the library chose the one of them in which the uppermost soldier was on a lower stair.