We have a sequence of n push/pop operations.
It is evident that resizing the array is an expensive operation.
The worst-case is when the maximum number of resizes happen, which happens when each of the n operations are push operations.
Let us look at how STACKSIZE grows.
Initially, it is $2^0 - 1$.
When the array is resized, if original size is $2^i - 1$, then in the next step it is $2[2^i - 1] + 1 = 2^{i+1} - 1$.
Clearly, we can prove by induction that STACKSIZE is $2^i - 1$ for some i at every stage.
Finally, if STACKSIZE = $2^k - 1$, then $(2^k - 1 \geq n)$ must hold.
Precisely, k is the smallest integer such that $2^k - 1 \geq n$
$[k = \lceil \log_2(n+1) \rceil]$.
Also, notice that k is the number of resizes done as well.
Now, to resize array from $2^i - 1$ to $2^{i+1} - 1$, we need to perform $2^i - 1$ pops and $2^i - 1$ pushes.
Hence it takes $2(2^i - 1)$ steps.
So total number of extra steps $\leq \sum_{i=0}^{k-1} 2(2^i - 1) = 2[2^k - 1 - k]$
for a looser estimate,
$$\sum_{i=0}^{k-1} 2(2^i - 1) < \sum_{i=0}^{k-1} 2^{i+1} = 2(2^k - 1) < 2^{k+1} < 4n \quad (\text{as } k = \lceil \log_2(n+1) \rceil)$$
$\therefore$ total number of steps doesn't exceed $5n$ [taking push & pop as elementary]
Or, if n push & pop operations using a regular stack involves $f(n)$ steps, this new stack uses at most $5f(n)$ steps.

[SIMILAR ANALYSIS FOR QUEUE]