Practical Analysis

Assume array *A* is indexed from 1 to *n*.

INEFFICIENT SORT(*A*, *n*)

1. **for** *i* = 1 **to** *n*! **do**
2. Boolean *sortedSoFar* = TRUE;
3. *j* = 1;
4. *P* = nextPermutation(*A*); // theta(n)
5. **while** *j* < *n* **and** *sortedSoFar* **do**
6. **if** *P*[*j*] > *P*[*j* + 1]
7. **then** *sortedSoFar* = FALSE // O(n)
8. *j*++
9. **if** (*sortedSoFar*) **then output** *P* //theta(n)

Analyze the worst-case complexity of INEFFICIENT SORT assuming that the nextPermutation function always takes $\Theta(n)$ time.

Complexity: theta(n. n!) = theta(n!)

lim(n!/(n n!) = lim(1/n) = 0

Your answer should fit above the line!