Dada a lista de números inteiros $\{5, 2, 8, 3, 1, 6, 4\}$, vamos ordená-la em ordem crescente usando a ordenação por inserção.

A ordenação por inserção é um algoritmo que funciona da seguinte maneira:

1. Comece com a primeira elemento da lista: 5
2. Compare o elemento atual (5) com o próximo elemento da lista (2). Como $5 > 2$, troque-os de posição: $\{2, 5, 8, 3, 1, 6, 4\}$
3. Compare o elemento atual (5) com o próximo elemento da lista (8). Como $5 < 8$, não há necessidade de trocar.
4. Compare o elemento atual (8) com o próximo elemento da lista (3). Como $8 > 3$, troque-os de posição: $\{2, 5, 3, 8, 1, 6, 4\}$
5. Compare o elemento atual (8) com o próximo elemento da lista (1). Como $8 > 1$, troque-os de posição: $\{2, 5, 3, 1, 8, 6, 4\}$
6. Compare o elemento atual (8) com o próximo elemento da lista (6). Como $8 > 6$, troque-os de posição: $\{2, 5, 3, 1, 6, 8, 4\}$
7. Compare o elemento atual (8) com o próximo elemento da lista (4). Como $8 > 4$, troque-os de posição: $\{2, 5, 3, 1, 6, 4, 8\}$
8. Compare o elemento atual (6) com o próximo elemento da lista (4). Como $6 > 4$, troque-os de posição: $\{2, 5, 3, 1, 4, 6, 8\}$
9. Compare o elemento atual (5) com o próximo elemento da lista (3). Como $5 > 3$, troque-os de posição: $\{2, 3, 5, 1, 4, 6, 8\}$
10. Compare o elemento atual (5) com o próximo elemento da lista (1). Como $5 > 1$, troque-os de posição: $\{2, 3, 1, 5, 4, 6, 8\}$
11. Compare o elemento atual (3) com o próximo elemento da lista (1). Como $3 > 1$, troque-os de posição: $\{2, 1, 3, 5, 4, 6, 8\}$
12. Compare o elemento atual (2) com o próximo elemento da lista (1). Como $2 > 1$, troque-os de posição: $\{1, 2, 3, 5, 4, 6, 8\}$
13. Compare o elemento atual (5) com o próximo elemento da lista (4). Como $5 > 4$, troque-os de posição: $\{1, 2, 3, 4, 5, 6, 8\}$

A lista ordenada é: $\{1, 2, 3, 4, 5, 6, 8\}$

Essa é a solução para o problema usando a ordenação por inserção. Note que a complexidade temporal da ordenação por inserção é $O(n^2)$, o que significa que o tempo de execução aumenta rapidamente com o tamanho da entrada.

As citações fornecidas mencionam a importância da eficiência algorítmica e como a escolha do algoritmo certo pode afetar significativamente o desempenho. No exemplo acima, a ordenação por inserção foi usada para resolver um problema de ordenação, mas existem outros algoritmos mais eficientes, como a ordenação por intercalação, que podem ser usados dependendo do tamanho da entrada e dos recursos disponíveis.