Algoritmos de ordenação
Seleção, Inserção e Flutuação

Sumário

Problema de ordenação

Entrada

Uma sequência de números $\langle a_1, a_2, \ldots, a_n \rangle$.

Saída

Uma permutação (reordenação) $\langle a'_1, a'_2, \ldots, a'_n \rangle$ da sequência de entrada tal que $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

Ordenação por inserção

```
insertion-sort(A, n)
1  for i = 1 to n
2     chave = A[i]
3     j = i - 1
4     while j > 0 e A[j] > chave
5       A[j + 1] = A[j]
6       j = j - 1
7     A[j + 1] = chave
```

Análise do insertion-sort

Melhor caso

Ocorre quando A já esta ordenado. O laço da linha 1 é executado $n+1$ vezes. O laço da linha 4 é executado uma vez para cada valor de i (a condição `A[j] > chave` sempre é falsa, porque A já está ordenado), portanto o laço da linha 4 é executado n vezes. Logo, o tempo de execução do algoritmo é $n+1+n = O(n)$.

Pios caso

Ocorre quando A está invertido. Neste caso o laço da linha 4 nunca é interrompido pela condição `A[j] > chave` (pois a está invertido e chave deve ser levada para a primeira posição). Desta forma o laço da linha 4 é executado $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$. Logo, o tempo de execução do algoritmo é $O(n^2)$.

Caso médio

Ocorre quando os valores de A estão distribuídos aleatoriamente. Neste caso, a chave deverá ser levada na média, até a metade do vetor já ordenado, isto é, na posição $\frac{i}{2}$. Portanto, o laço da linha 4 será executado $\sum_{i=1}^{n} \frac{i}{2} = \frac{n(n+1)}{4}$. O que não muda o limite do pior caso. Logo, o tempo de execução do algoritmo é $O(n^2)$.

Ordenação por seleção

```
selection-sort(A, n)
1  for i = 1 to n
2    min = i
3    for j = i + 1 to n
4      if A[j] > A[min]
5        min = j
6    troca A[i], A[min]
```

Análise do selection-sort

- Como nenhum dos laços depende dos valores de A, apenas de n, não existe distinção entre pior caso, melhor caso e caso médio.
- O laço da linha 1 é executado $n+1$ vezes. O laço da linha 3 é executado $\sum_{i=1}^{n} n - i = \frac{n(n+1)}{2} - n$ vezes. Portanto, o tempo de execução do algoritmo é $n + 1 + \frac{n(n+1)}{2} - n = O(n^2)$.

Ordenação por flutuação (Bubblesort)

```
bubble-sort(A, n)
1  trocado = True
2  while trocado
3    trocado = False
4    for i = 1 to n - 1
5      if A[i] > A[i+1]
6        troca A[i], A[i+1]
7        trocado = True
```

Análise do bubble-sort

Melhor caso

Ocorre quando A já esta ordenado. A condição da linha 5 nunca será verdadeira, porque A está ordenado, o que implica que a linha 7 não será executada e o laço da linha 2 não será repetido. Desta forma, o laço da linha 4 será executado n vezes. Logo, o tempo de execução do algoritmo é $O(n)$.

Pios caso

Ocorre quando A está invertido. Na primeira iteração do algoritmo, o laço da linha 4 é executado n vezes, o maior elemento do vetor é "levado" para a posição n. Na segunda iteração, o laço da linha 4 é executado novamente n vezes, e o segundo maior elemento é levado para a posição $n-1$. O algoritmo segue desta maneira até que o vetor esteja ordenado. Como em cada iteração do algoritmo, apenas um elemento é colocado em sua posição correta, são necessárias n iterações, como em cada iteração o laço da linha 4 é executado n vezes, temos que o tempo de execução do algoritmo é $n.n = O(n^2)$.

Caso médio

Ocorre quando os valores de A estão distribuídos aleatoriamente. A análise é semelhante ao do pior caso, e o tempo de execução do algoritmo é o mesmo $O(n^2)$.