Universidade de Brasília
Departamento de Ciência da Computação
Projeto e Análise de Algoritmos

Atividade 02

Selection Sort

José Antônio Alcântara da Silva de Andrade Mat: 232013031

Professor:
Flávio Leonardo Calvacanti de Moura
Turma 02

Brasília, DF

7 de abril de 2025

Selection Sort

A solução a seguir descreve uma versão não-recursiva e memória-eficiente do algoritmo de ordenação *selection sort*, e realiza sua correção usando-se do método das invariâncias.

Definição

O algoritmo de *selection sort* consiste na procura pelo menor termo de uma lista e, uma vez encontrado, adicionando-o à última posição da seção ordenada da lista. Um pseudo-código demonstrando esse comportamento para uma lista arbitrária `A[1...n]` pode ser visto na Listagem 1.

Listagem 1: Pseudo-código do Selection Sort

```
FOR i = 1 TO n {
    min_idx <- i;
    FOR j = i + 1 TO n {
        IF A[j] < A[min_idx] {
            min_idx <- j;
        }
    }
    IF min_idx != i {
        aux <- A[i];
        A[i] <- A[min_idx];
        A[min_idx] <- A[aux];
    }
}
```

O código consiste de dois loops: o loop interno que procura o índice do menor termo da lista não-ordenada, e o loop externo que adiciona à lista ordenada o próximo menor termo. Realizar-se-á a prova de correção de ambos ciclos.

Correção Interna

Para a correção interna, toma-se esta invariância: "Antes de cada iteração, o valor de `min_idx` é o índice do menor termo da sublista `A[i...j-1]`."

Durante a inicialização do ciclo, ou seja, quando as variáveis são `i` e `j = i + 1`, teremos a sublista `A[i...j-1] = A[i...i]`, que se reduz para apenas `A[i]`. Como a sublista `A[i]` consiste de apenas um termo, e o valor de `min_idx`, nesse instante, é `i`, temos garantidamente que a invariância ocorre.

Ao final do ciclo, teremos que, se o valor de `A[j]` for menor do que `A[min_idx]` (o atual menor termo), troca-se o valor de `min_idx` pelo valor de `j`. Dessa forma, ao atingir o fim do ciclo, a sublista `A[i...j]` estará ordenada, provando-se a manuntenção da invariância.

Por fim, o loop termina naturalmente quando `j = n`, indicando que, pela manuntenção da invariância, o valor `min_idx` representa o índice do menor item da sublista `A[i...n]`. Ao sair do ciclo, `min_idx` é, garantidamente, o índice do menor termo da sublista `A[i...n]`.

Veremos na correção do loop externo que essa sublista é a parte não-ordenada do vetor, garantindo, então, que encontramos o índice do próximo menor item.

Correção Externa

Para a correção externa, toma-se esta invariância: "Antes de cada iteração, a sublista `A[1...i-1]` está ordenada."

Durante a inicialização do ciclo, quando a variável `i = 1`, teremos a sublista `A[1..0]`, que equivale-se à lista sem itens. Por definição, a lista sem itens é ordenada, ou seja, a invariação é válida.

Ao final do ciclo, verifica-se se o termo `min_idx` é diferente do valor atual de `i`. Se isso não ocorrer, significa que os valores são iguais, ou seja, `i` é o índice do menor termo na sublista `A[i...n]`, não é necessário realizar outras ações. Se isso ocorrer, os valores são diferentes, então realiza-se uma troca de posição: o valor no índice `i` é trocado de valor com o índice `min_idx`. Ambos casos são verificados pela correção do loop interno, e no final de ambos, a sublista `A[1...i]` está garantidamente ordenada, ocorrendo manuntenção da invariância para o próximo ciclo.

Por fim, o ciclo termina naturalmente quando `i = n` e, pela verificação do processo de manuntenção da invariância, será garantido que a sublista `A[1...i]` é válida. Como `i = n`, teremos que a sublista `A[1...n] = A` agora se encontra ordenada.