Resumo: AEDS2 - prova 01

Ary

15/03/2025

1 Algoritmo de Ordenação por Seleção

A ordenação interna ocorre quando os elementos a serem ordenados cabem na memória principal.

O algoritmo segue os seguintes passos:

1. Procura-se o menor elemento do array.
2. Troca-se esse elemento com o primeiro elemento do array.
3. Repete-se o processo para o restante do array até que a ordenação esteja completa.

1.1 Exemplo

Dado o array inicial:

`[101, 115, 30, 63, 47, 20]`

Após cada iteração:

`[20, 115, 30, 63, 47, 101]`

`[20, 30, 115, 63, 47, 101]`

`[20, 30, 47, 63, 115, 101]`

`[20, 30, 47, 63, 115, 101]`

`[20, 30, 47, 63, 101, 115]`

1.2 Implementação em Pseudocódigo

```
for (int i = 0; i < (n - 1); i++) {
    int menor = i;
    for (int j = (i + 1); j < n; j++) {
        if (array[menor] > array[j]) {
            menor = j;
```

```
        }
    }
    swap(array[i], array[menor]);
}
```

1.3 Análise de Complexidade

A complexidade do algoritmo de ordenação por seleção, em termos de comparações e movimentações, é:

- **Melhor caso**: $O(n^2)$
- **Pior caso**: $O(n^2)$
- **Caso médio**: $O(n^2)$

O número de comparações pode ser expresso como a soma:

$$\sum_{i=0}^{n-2}(n-1-i) = \frac{(n-1)n}{2} \Rightarrow O(n^2) \tag{1}$$

Sendo que o loop interno repete $n-(i+1)$ para $n-1$.
O número de trocas é limitado a $O(n)$.

1.4 Conclusão

O algoritmo de ordenação por seleção é simples e fácil de implementar, mas não é eficiente para grandes conjuntos de dados devido à sua complexidade quadrática.

2 Somatórios

2.1 Definição

$$\sum_{i=a}^{b} f(i) = f(a) + f(a+1) + \cdots + f(b) \tag{2}$$

2.2 Regras Básicas de Manipulação

- **Distributividade**: $\sum cf(i) = c\sum f(i)$
- **Associatividade**: $\sum(f(i)+g(i)) = \sum f(i) + \sum g(i)$
- **Comutatividade**: A ordem dos termos pode ser alterada sem mudar o resultado.

2.3 Progressão Aritmética (PA)

Uma PA é uma sequência cuja razão (diferença) entre dois termos consecutivos é constante. Por exemplo, 5, 7, 9, 11, 13,

Cada termo da PA será a $i = a + b.i$, onde a é o termo inicial; b, a razão; e, i, a ordem do termo.

Na sequência acima, a e b são iguais a 5 e 2, respectivamente. Logo, temos:

$$(5 + 2.0), (5 + 2.1), (5 + 2.2), (5 + 2.3), (5 + 2.4), \; ... \tag{3}$$

Aplique as regras de transformação para obter a fórmula fechada da soma S_n dos elementos de uma Progressão Aritmética (PA):

$$\begin{aligned}
s_n &= \sum_{i=0}^{n}(a + bi) \\
&\textit{Aplicando a Comutatividade} \\
&= \sum_{i=0}^{n}[a + b(n - i)] = \sum_{i=0}^{n}[a + bn - bi)] \\
2s_n &= \sum_{i=0}^{n}(a + bi) + \sum_{i=0}^{n}[a + bn - bi)] \\
&\textit{Aplicando a Associatividade} \\
&= \sum_{i=0}^{n}[a + \cancel{bi} + a + bn - \cancel{bi})] \\
2s_n &= \sum_{i=0}^{n}[2a + bn] \\
&\textit{Aplicando a Distributividade} \\
&= (2a + bn)\sum_{i=0}^{n} 1 \\
s_n &= \frac{(2a + bn)(n + 1)}{2}
\end{aligned} \tag{4}$$

Fórmula fechada para o somatório de Gauss, usando a fórmula da soma de uma progressão aritmética qualquer:

$$\begin{aligned}
\sum_{i=0}^{n} i &= 0+1+2+3+\cdots+n \\
s_n &= \sum_{i=0}^{n}(0+1i) \qquad \begin{matrix}\text{a} = i_0 = 0 \\ \text{b} = \text{r} = 1\end{matrix} \\
&= \frac{(\cancel{2\times 0}+1n)(n+1)}{2} \\
&= \frac{n(n+1)}{2}
\end{aligned} \tag{5}$$

2.4 Propriedades Somatorio

2.4.1 P1 - Combinando Conjuntos

Combina conjuntos de índices diferentes. No caso, se I e I' são dois conjuntos quaisquer de inteiros, então:

$$\sum_{i\in I} a_i + \sum_{i\in I'} a_i = \sum_{i\in I\ I\cup I'} a_i + \sum_{i\in I\ I\cap I'} a_i \tag{6}$$

Se $A = 1,2,3$ e $B = 3,5,7$, então $A \cup B = 1,2,3,5,7$ e $A \cap B = 3$

2.4.2 P2 - Base para a Perturbação

Dada uma soma genérica qualquer $\sum_{0\le i\le n} a_i$ temos que

$$s_{n+1} = a_0 + a_1 + a_2 + \cdots + a_n + a_{n+1}$$

1ª Forma

$$s_{n+1} = s_n + a_{n+1}$$

2ª Forma

$$s_{n+1} = \sum_{0\le i\le n+1} a_i = a_0 + \sum_{1\le i\le n+1} a_i = a_0 + \textstyle\sum_{1\le i\le n} a_{i+1} \tag{7}$$

$$s_n + a_{n+1} = a_0 + \sum_{1\le i\le n} a_{i+1}$$

Exemplo 1:

$$
\begin{aligned}
s_n &= \sum_{0}^{n} ax^i \\
s_{n+1} &= s_n + ax^{n+1} = ax^0 + \sum_{1}^{n} ax^{i+1} \\
s_n + ax^{n+1} &= ax^0 + \sum_{1}^{n} ax^{i+1} \\
s_n + ax^{n+1} &= ax^0 + a\sum_{1}^{n} x^1 x^i \\
s_n + ax^{n+1} &= ax^0 + x^1 a\sum_{1}^{n} x^i \\
s_n + ax^{n+1} &= a \times 1 + x\sum_{1}^{n} ax^i \\
s_n + ax^{n+1} &= a + xs_n \\
s_n + \cancel{ax^{n+1}} - xs_n \cancel{-ax^{n+1}} &= a + \cancel{xs_n} - ax^{n+1} - \cancel{xs_n} \\
s_n - xs_n &= a - ax^{n+1} \\
s_n - xs_n &= a - ax^{n+1} \\
(1-x)s_n &= a - ax^{n+1} \\
s_n &= \frac{a - ax^{n+1}}{1-x} \quad \textit{para } x \neq 1
\end{aligned} \tag{8}
$$

Exemplo 2:

$$
\begin{aligned}
s_n &= \sum_0^n i2^i \\
s_n + (n+1)2^{n+1} &= 0 + \sum_0^n (i+1)2^{i+1} \\
s_n + (n+1)2^{n+1} &= 0 + \sum_0^n i2^{i+1} + \sum_0^n 2^{i+1} \\
s_n + (n+1)2^{n+1} &= 0 + 2\sum_0^n i2^i + \sum_0^n 2^{i+1} \\
s_n + (n+1)2^{n+1} &= 2s_n + 2\sum_0^n 2^i \qquad (9) \\
s_n + (n+1)2^{n+1} &= 2s_n + 2\frac{1 \times 2^{n+1}}{-1} \\
s_n + (n+1)2^{n+1} &= 2s_n + 2(2^{n+1} - 1) \\
(n+1)2^{n+1} - 2(2^{n+1} - 1) &= 2s_n - s_n \\
s_n &= (n+1)2^{n+1} - 2(2^{n+1} - 1) \\
&= n2^{n+1} - 2^{n+1} - 4^{n+1} + 2 \\
&= n2^{n+1} - 2^{n+1} + 2 \\
\sum_0^n i2^i &= (n-1)2^{n+1} + 2
\end{aligned}
$$

2.5 Adivinhe a Resposta, Prove por Indução

2.5.1 1º Passo (passo base)

Provar que a fórmula é verdadeira para o primeiro valor, substituindo n na equação pelo primeiro valor

2.5.2 2º Passo (indução propriamente dita)

Supondo que $n > 0$ e que a fórmula é válida quando substituímos n por $(n-1)$.

$$s_n = s_{n-1} + a_n$$

Pove por indução que a fórmula abaixo para a soma dos quadrados perfeitos é verdadeira:

$$\sum_{0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \quad \text{para } n \geq 0$$

1º Passo

$$\sum_{0}^{n} i^2 = \frac{0(0+1)(20+1)}{6}$$
$$= \frac{0}{6} = 0$$

2º Passo

$$\frac{n(n+1)(2n+1)}{6} = \frac{(n-1)((n-1)+1)(2(n-1)+1)}{6} + a_n \tag{10}$$
$$\frac{(n^2+n)(2n+1)}{6} = \frac{(n-1)(n)(2n-1)}{6} + n^2$$
$$\frac{2n^3+n^2+2n^2+n}{6} = \frac{(n^2-n)(2n-1)}{6} + n^2$$
$$\frac{2n^3+3n^2+n}{6} = \frac{2n^3-n^2-2n^2+n}{6} + n^2$$
$$= \frac{[2n^3-n^2-2n^2+n]+6n^2}{6}$$
$$\frac{2n^3+3n^2+n}{6} = \frac{2n^3+3n^2+n}{6}$$

Encontre a fórmula fechada do somatório abaixo e, em seguida, prove a usando indução matemática:

$$\begin{aligned}
\sum_{0}^{n} 3+i &= 3(n+1) + \frac{n(n+1)}{2} \\
&= \frac{6n+6+n(n+1)}{2} \\
&= \frac{6n+6+n^2+n}{2} \\
&= \frac{n^2+7n+6}{2}
\end{aligned}$$

$$\begin{aligned}
&\textbf{1}^{\textbf{o}} \textbf{ Passo} \\
\frac{0^2+7\times 0+6}{2} &= 3 \\
\frac{0+0+6}{2} &= 3 \\
3 &= 3 \\
&\textbf{2}^{\textbf{o}} \textbf{ Passo} \\
\frac{n^2+7n+6}{2} &= \frac{(n-1)^2+7(n-1)+6}{2} + a_n \\
\frac{n^2+7n+6}{2} &= \frac{(n^2-2n+1)+(7n-7)+6}{2} + (3+n) \\
&= \frac{(n^2-2n+1)+(7n-7)+6+(6+2n)}{2} \\
\frac{n^2+7n+6}{2} &= \frac{n^2+7n+6}{2}
\end{aligned} \tag{11}$$

Encontre a fórmula fechada do somatório abaixo e, em seguida, prove a usando indução matemática:

$$
\begin{aligned}
\sum_1^n [(2i+1)^2 - (2i)^2] &= \sum_1^n [(4i^2 + 4i + 1) - (2i)^2] \\
&= \sum_1^n [4i^2 + 4i + 1 - 4i^2] \\
&= \sum_1^n [4i + 1] \\
&= 4\sum_1^n i + \sum_1^n 1 \\
&= 4\frac{n(n+1)}{2} + n \\
&= 2n(n+1) + n \\
&= 2n^2 + 3n
\end{aligned} \tag{12}
$$

1º Passo

$$
\begin{aligned}
2 \times 1^2 + 3 \times 1 &= [(2 \times 1 + 1)^2 - (2 \times 1)^2] \\
2 + 3 &= [(3)^2 - 2^2] \\
5 &= 9 - 4 \\
5 &= 5
\end{aligned}
$$

2º Passo

$$
\begin{aligned}
2n^2 + 3n &= 2(n-1)^2 + 3(n-1) + a_n \\
2n^2 + 3n &= 2(n-1)^2 + 3(n-1) + [(2n+1)^2 - (2n)^2] \\
&= 2(n^2 - 2n + 1) + 3n - 3 + [4n^2 + 4n + 1 - 4n^2] \\
&= 2n^2 - 4n + 2 + 3n - 3 + [4n + 1] \\
&= 2n^2 + 3n
\end{aligned}
$$

Perturbe o somatório dos cubos para encontrar a fórmula fechada do somatóriodos quadrados:

$$
\begin{aligned}
s_n &= \sum_{0}^{n} i^2 \\
scubo_n &= \sum_{0}^{n} i^3 \\
scubo_n + (n+1)^3 &= 0^3 + \sum_{0}^{n} (i+1)^3 \\
scubo_n + (n+1)^3 &= 0^3 + \sum_{0}^{n} [i^3 + 3i^2 + 3i + 1] \\
scubo_n + (n+1)^3 &= \sum_{0}^{n} i^3 + \sum_{0}^{n} 3i^2 + \sum_{0}^{n} 3i + \sum_{0}^{n} 1 \\
scubo_n + (n+1)^3 &= scubo_n + 3s_n + \frac{3n(n+1)}{2} + (n+1) \\
(n+1)^3 &= 3s_n + \frac{3n(n+1)}{2} + (n+1) \\
(n+1)^3 &= \frac{6s_n + 3n(n+1) + 2(n+1)}{2} \\
2(n+1)^3 - 3n(n+1) - 2(n+1) &= 6s_n \\
6s_n &= 2n^3 + 6n^2 + 6n + 2 - 3n^2 - 3n - 2n - 2 \\
6s_n &= 2n^3 + 3n^2 + n \\
s_n &= \frac{2n^3 + 3n^2 + n}{6}
\end{aligned}
\tag{13}
$$

2.5.3 Faça um método int somatorioPA(double a, double b, int n) que retorna o somatório dos n primeiros termos de uma PA com termo inicial a e razão b.

```
int somatorioPA(double a, double b, int n) {
    return (2 * a + b * n) * (n + 1) / 2;
}
```

2.5.4 Um algoritmo de ordenação tradicional é o Inserção. Faça a análise decomplexidade desse algoritmo para os números de comparações e movimentações entre registros no pior e melhor caso

```
void insertion(int *arr, int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
```

```
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
```

$$
\begin{aligned}
&\textit{Laço interno:}\\
&I(i-1) = \sum_{0}^{i-1} 1 = i\\
&\textit{Laço Externo:}\\
&T(n-1) = \sum_{0}^{n-1} I(i-1) = \sum_{0}^{n-1} i = \frac{(n-1)(n)}{2}\\
&\textit{Melhor caso:}\\
&C(n) = (n-1) = \Theta(n)\\
&\textit{Pior caso:}\\
&C(n) = \frac{(n-1)(n)}{2} = \Theta(n^2)\\
&Mi(n) = C(n) - 1\\
&M(n) = Mi(n) + 2
\end{aligned}
\tag{14}
$$

https://www.cs.umd.edu/users/meesh/cmsc351/mount/lectures/lect3-sums-and-loops.pdf

3 Estruturas de dados básicas lineares

3.1 Lista linear

Tipo Abstrato de Dados (TAD) no qual podemos inserir e remover elementos em qualquer posição. Composta por um **Array** (de elementos) e um **n** (contador). A remoção é logica.

```
void inserirInicio(int x) {
    if (n >= MAXTAM)
        exit(1);
    //levar elementos para o fim do array
    for (int i = n; i > 0; i--){
        array[i] = array[i-1];
    }
    array[0] = x;
    n++;
}
```

```
void inserir(int x, int pos) {
    if (n >= MAXTAM || pos < 0 || pos > n)
        exit(1);
    //levar elementos para o fim do array
    for (int i = n; i > pos; i--){
        array[i] = array[i-1];
    }
    array[pos] = x;
    n++;
}

void inserirFim(int x) {
    if (n >= MAXTAM)
        exit(1);
    array[n] = x;
    n++;
}

int removerInicio() {
    if (n == 0)
        exit(1);
    int resp = array[0];
    n--;
    for (int i = 0; i < n; i++){
        array[i] = array[i+1];
    }
    return resp;
}

int remover(int pos) {
    if (n == 0 || pos < 0 || pos >= n)
        exit(1);
    int resp = array[pos];
    n--;
    for (int i = pos; i < n; i++){
        array[i] = array[i+1];
    }
    return resp;
}

int removerFim() {
    if (n == 0)
        exit(1);
    return array[--n];
}
```

3.2 Pilha (stack)

Primeiro elemento que entra é o último a sair. Pilha de prato.

First In Last Out (FILO)

Tem basicamente os métodos de inserir (empilhar, push) e remover (desempilhar, pop).

```
int array[];
int n;

void push(int x) {
    if (n >= MAXTAM)
        exit(1);
    array[n] = x;
    n++;
}

int pop() {
    if (n == 0)
        exit(1);
    return array[--n];
}
```

3.3 Fila (Queue)

Are you a FIFO? Because you are a Queue<T>. haha.

Primeiro elemento que entra é o primeiro a sair. Tem basicamente os métodos de inserir (enfileirar, enqueue) e remover (desenfileirar, dequeue). Fila de banco.

As formas de implementar são:

- **Inserir Final e Remover Início (IF e RI) (FIFO)**: A remoção não é eficiente.
- **Inserir Início e Remover Final (II e RF) (LIFO)**: A

Como implementar uma fila sem que uma das operações desloque todos os elementos?

R: Fazendo uma fila circular, ou seja, depois da última posição, retornamos para a primeira

```
class Fila {
    int[] array;
    int primeiro, ultimo;

    Fila (int tamanho) {
```

```
    array = new int[tamanho+1];
    primeiro = ultimo = 0;
}

void inserir(int x) throws Exception {
    if (((ultimo + 1) % array.length) == primeiro)
        throw new Exception("Erro!");
    array[ultimo] = x;
    ultimo = (ultimo + 1) % array.length;
}

int remover() throws Exception {
    if (primeiro == ultimo)
        throw new Exception("Erro!");
    int resp = array[primeiro];
    primeiro = (primeiro + 1) % array.length;
    return resp;
}

void mostrar() {
    int i = primeiro;
    System.out.print("[");
    while (i != ultimo) {
        System.out.print(array[i] + " ");
        i = (i + 1) % array.length;
    }
    System.out.println("]");
}
```