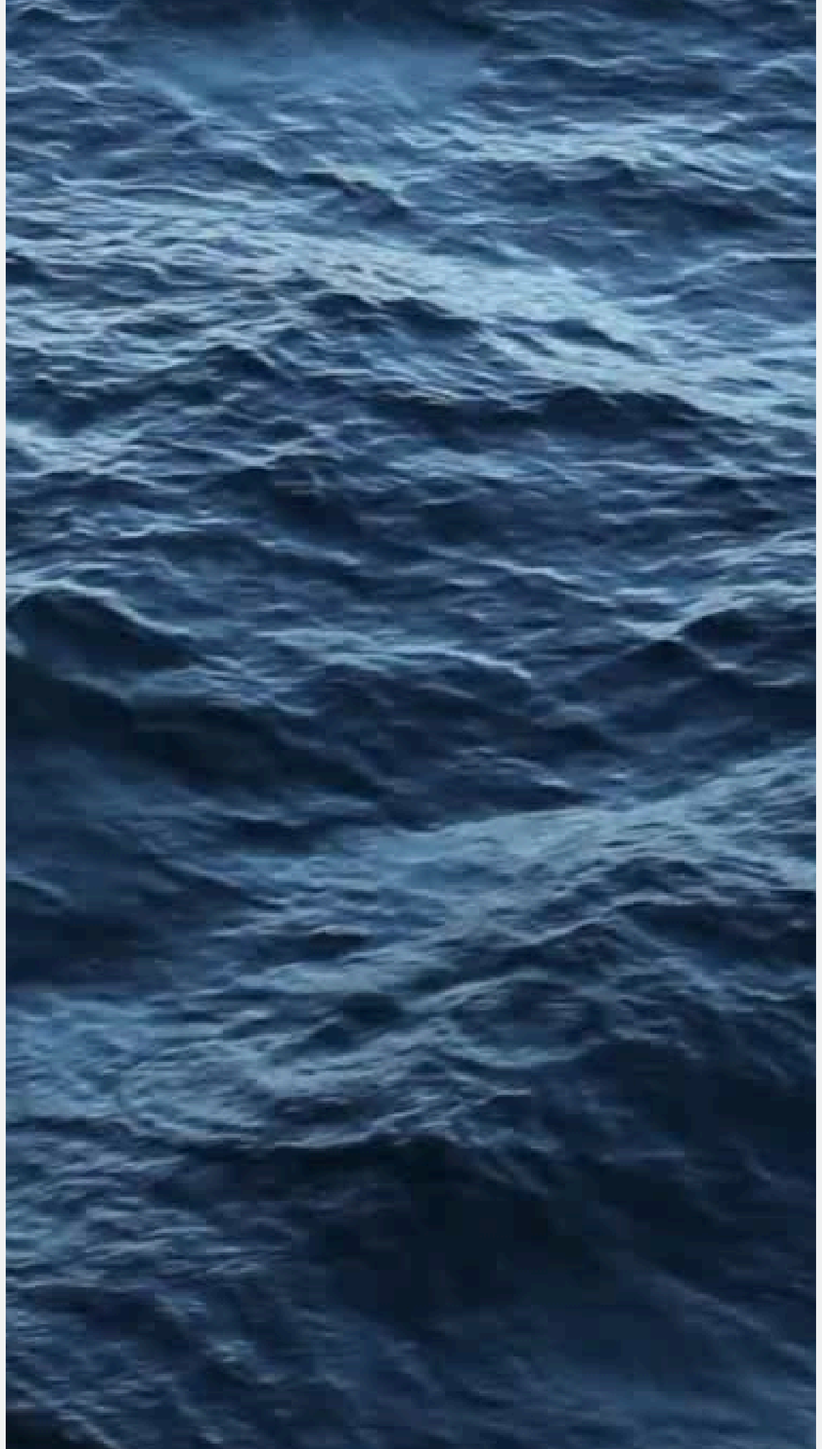
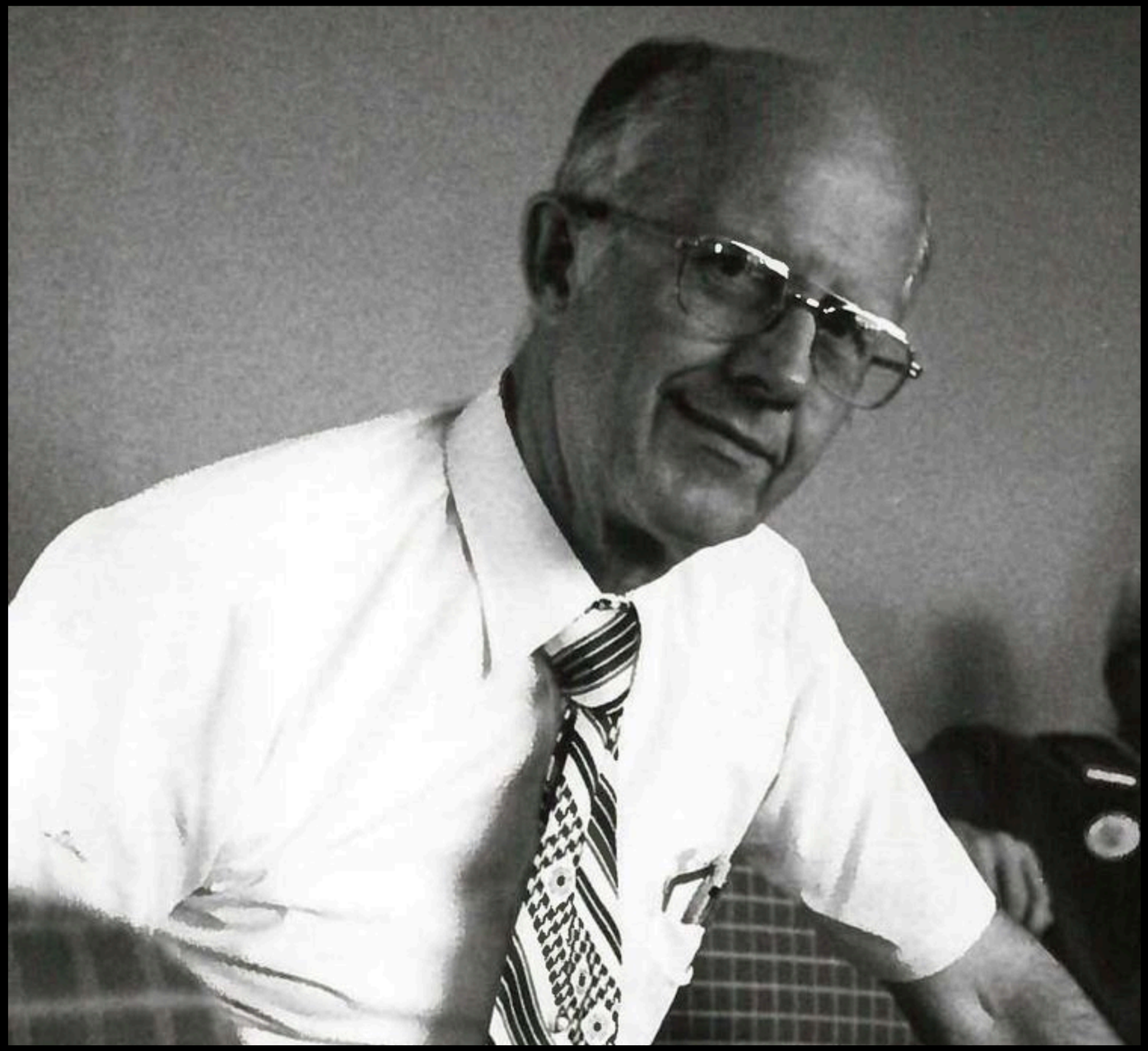
SHELL SORT



LUIZA DANIELLI, FABIANY MARINS E MAIKELEN PASQUALI

DONALD SHELL

O Shell sort foi criado por Donald Shell em 1959 e publicado pela Universidade de Cincinnati. É um algoritmo de ordenação eficiente, considerado uma generalização do ordenamento por inserção, que funciona ordenando elementos distantes uns dos outros e, gradualmente, diminuindo a distância até chegar à comparação de elementos adjacentes (distância de 1).



PONTOS POSITIVOS

Eficiência para dados de tamanho médio
- Mais rápido que Bubble Sort e Selection Sort em listas moderadas.

Otimização do Insertion Sort
- Reduz significativamente trocas e deslocamentos, especialmente nas fases finais.

Baixo consumo de memória (in-place)
- Ordena no próprio array, com complexidade de espaço O(1), ideal para sistemas com pouca memória.

Bom desempenho em dados quase ordenados
- Se a lista já estiver parcialmente ordenada, o Shell sort é ainda mais eficiente.

Simplicidade de implementação
- Mais fácil de programar do que algoritmos como Quicksort ou Mergesort.

Flexibilidade na sequência de espaçamento (gap)
- Permite ajustar o desempenho conforme as características dos dados.

PONTOS NEGATIVOS

Instabilidade

· não preserva a ordem de elementos iguais, o que pode causar problemas em algumas aplicações.

Complexidade de pior caso não ideal

· pode chegar a O(n^2) se a sequência de intervalos (gaps) for mal escolhida, tornando-o menos eficiente que Merge Sort ou Quick Sort.

Dependência da sequência de intervalos

· o desempenho depende fortemente da escolha dos gaps, e não há uma sequência universalmente ideal.

Menor eficiência em grandes volumes de dados

· costuma ser superado por algoritmos mais avançados em conjuntos muito grandes.

Complexidade na otimização

· exige estudo e implementação cuidadosa da sequência de gaps, o que torna a otimização mais trabalhosa.

Pouco usado atualmente

· menos comum em aplicações modernas, mas ainda útil para dados médios com baixa exigência de memória.

CÓDIGO

```
public class ShellSort
{
    2 referências
    public static void Sort(int[] arr)
    {
        int l = arr.Length;
        for (int gap = l / 2; gap > 0; gap /= 2)
        {
            for (int i = gap; i < l; i++)
            {
                int atual = arr[i];
                int j;
                for (j = i; j >= gap && arr[j - gap] > atual; j -= gap)
                {
                    arr[j] = arr[j - gap];
                }
                arr[j] = atual;
            }
        }
    }
```

CÓDIGO

```
        0 referências
        public static void Main()
        {
            int[] arr = { 64, 34, 25, 12, 22, 11, 90 };
            Console.WriteLine("\n\nVetor original");
            Console.WriteLine(string.Join(", ", arr));
            Sort(arr);
            Console.WriteLine("\n\nVetor ordenado com Shell Sort");
            Console.WriteLine(string.Join(", ", arr));
        }
}
```

CÓDIGO

```
Vetor original
64, 34, 25, 12, 22, 11, 90


Vetor ordenado com Shell Sort
11, 12, 22, 25, 34, 64, 90
```

OBRIGADA
PELA ATENÇÃO