DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Estruturas de Dados

ShellSort e Geração de Números Aleatórios

Professores: Marcio Santos
Wagner Meira Jr

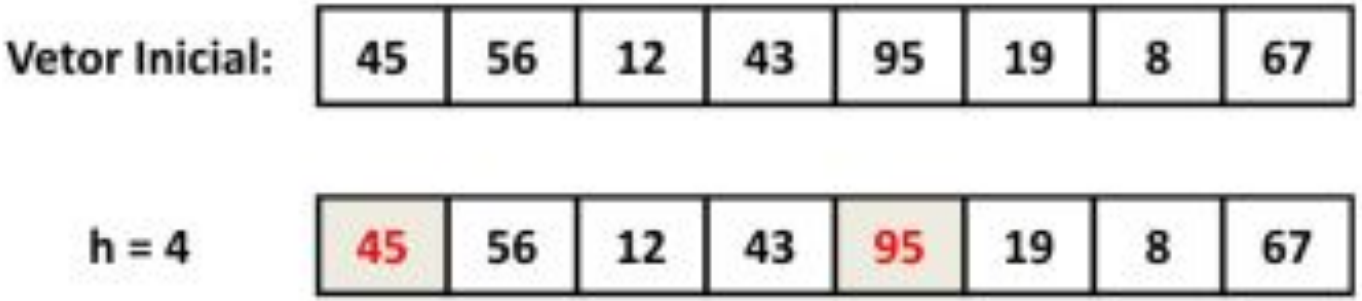
ShellSort - Ideia

- Proposto por Donald Shell em 1959
- É uma extensão do **InsertionSort**
- Problemas com o **InsertionSort**: sempre troca elementos vizinhos
- O método do Shell permite a troca de elementos distantes
- Trocamos itens separados por ***h*** posições.
- Vamos reduzindo o valor de h até chegar em 1. Quando o vetor estará ordenado.

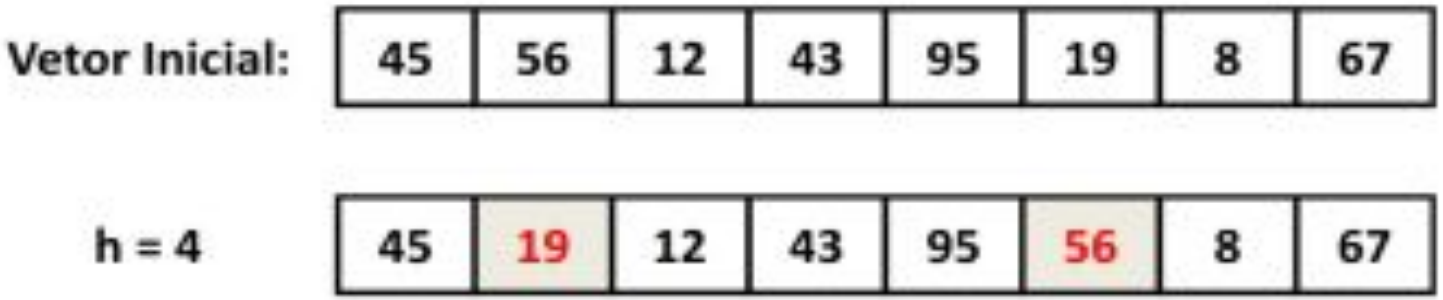
ShellSort - Execução

Vetor Inicial:	45	56	12	43	95	19	8	67

ShellSort - Execução

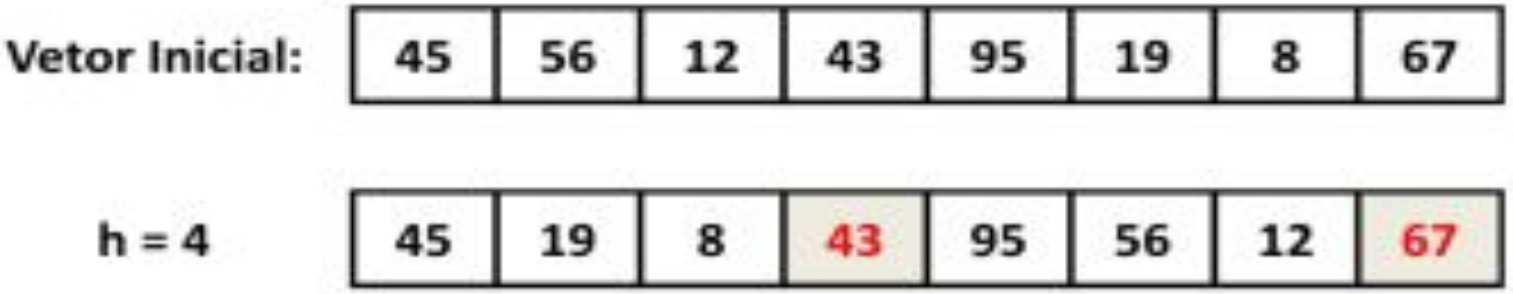


ShellSort - Execução



ShellSort - Execução

ShellSort - Execução

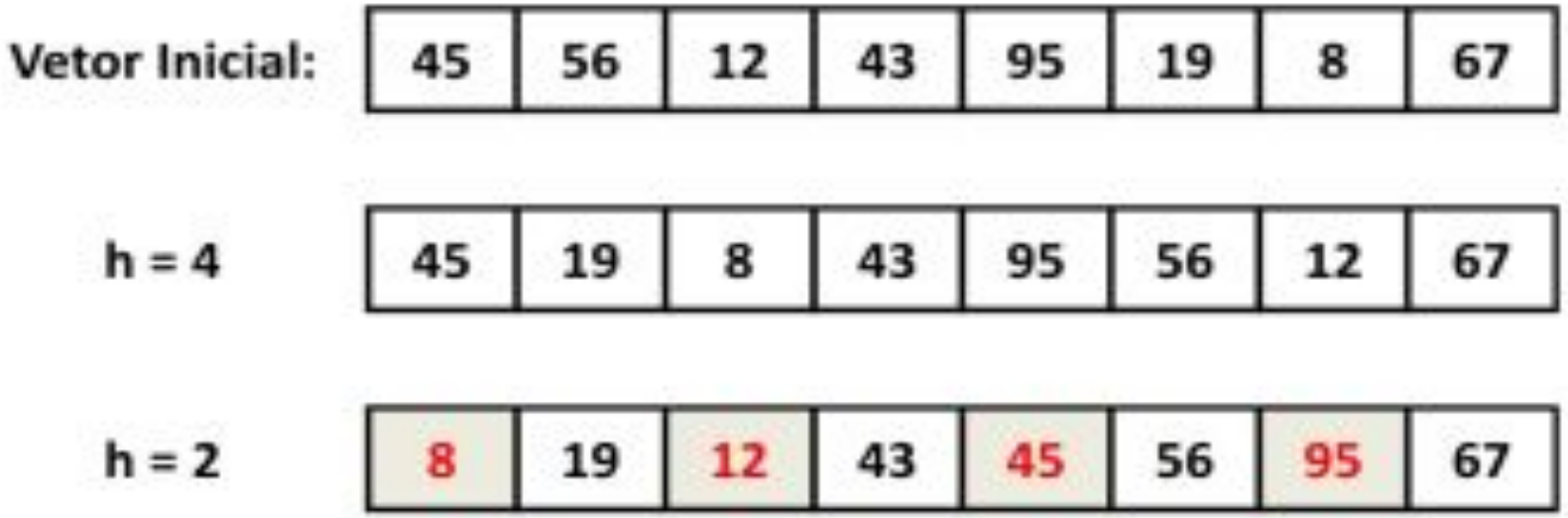


ShellSort - Execução

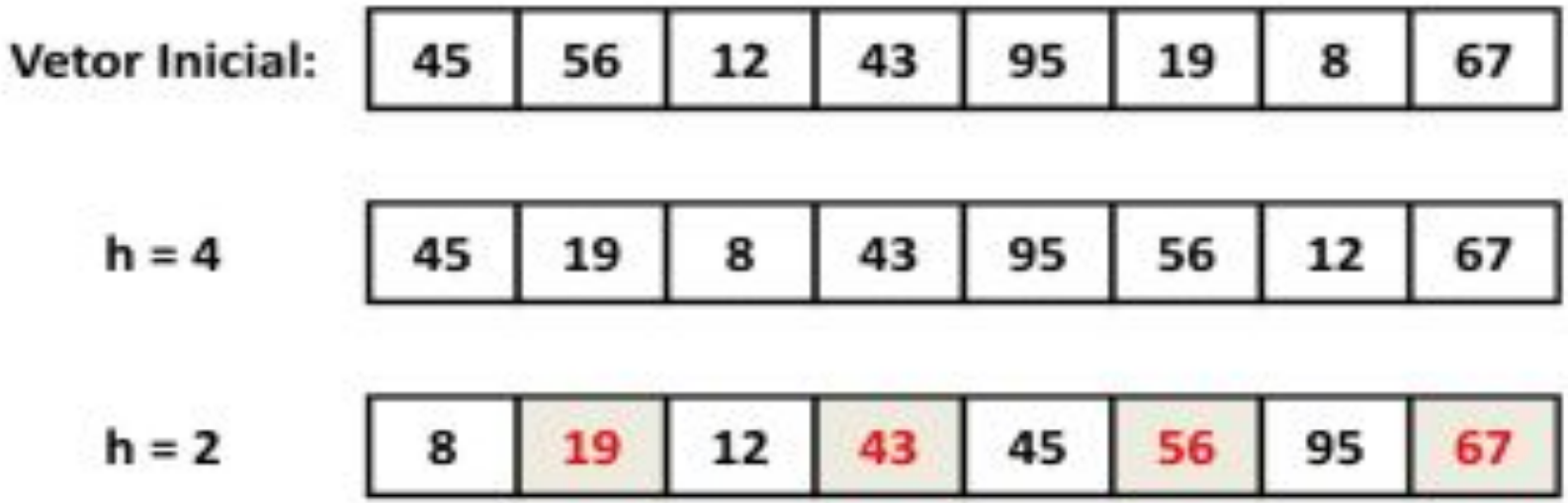
Vetor Inicial:	45	56	12	43	95	19	8	67

h = 4	45	19	8	43	95	56	12	67

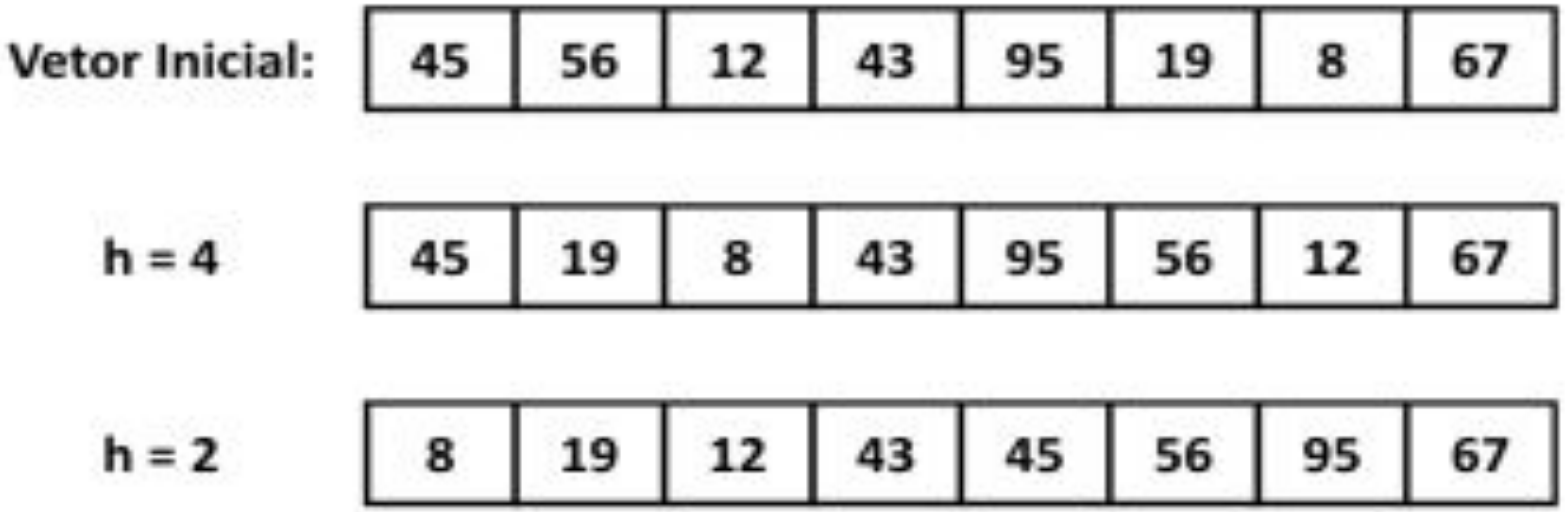
ShellSort - Execução



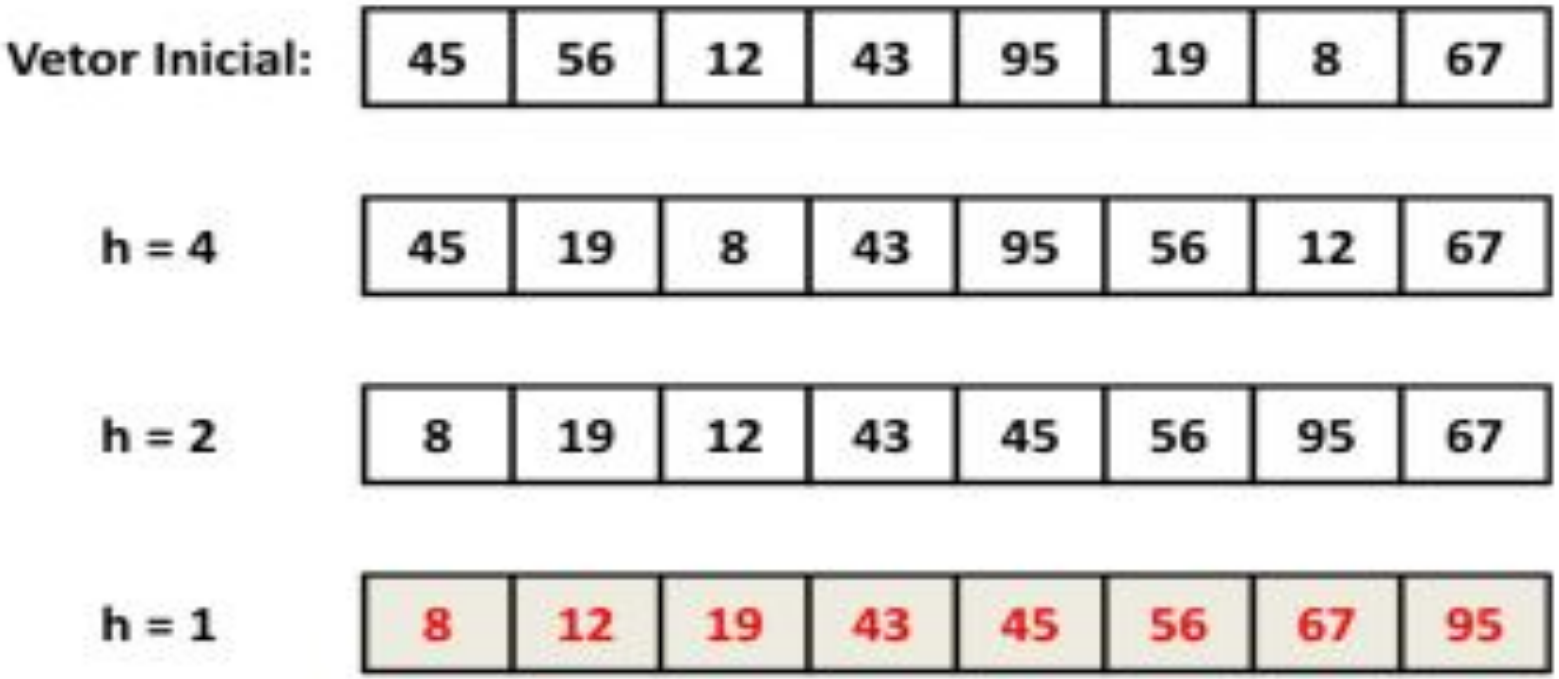
ShellSort - Execução



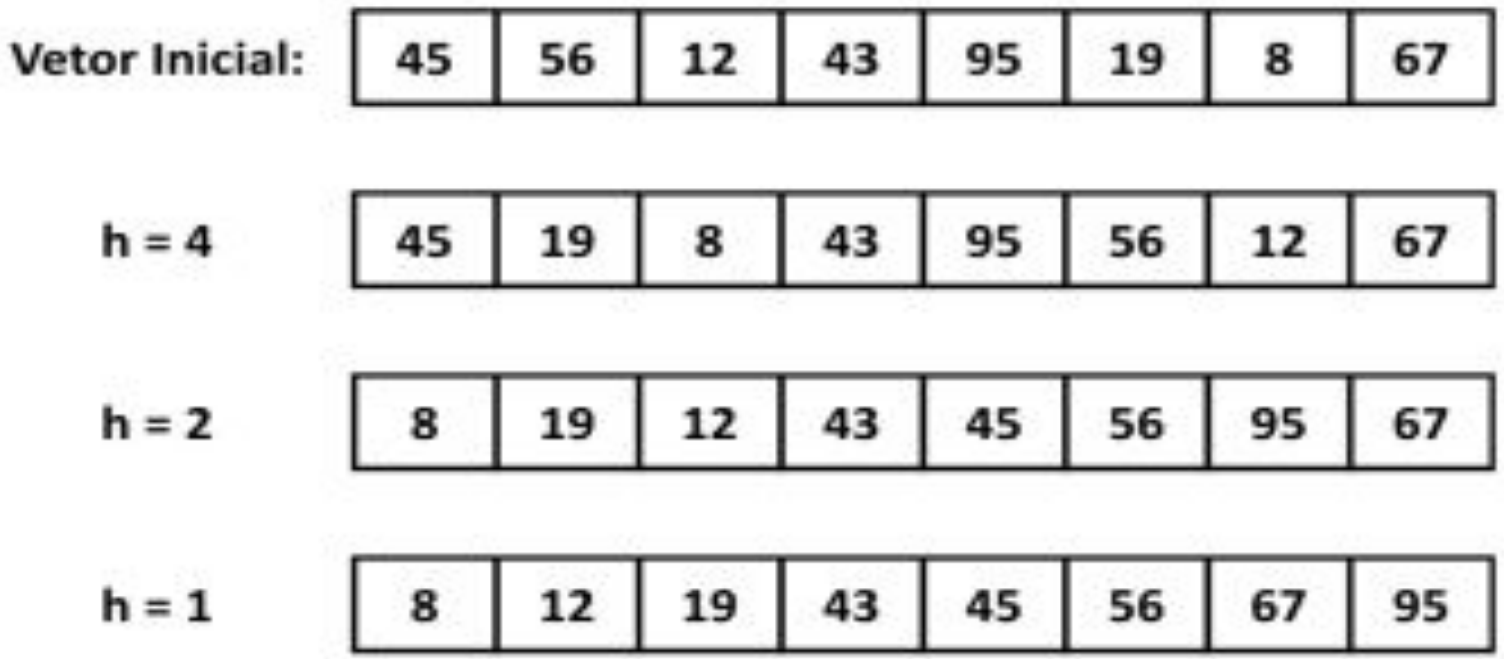
ShellSort - Execução



ShellSort - Execução



ShellSort - Execução



ShellSort - Pseudo Código

```
void shellSort(int *vet, int size) {
    int i, j, value;
    int h = 1;
    calcula valor de h;
    while (h > 0) {
        for(i = h; i < size; i++) {
            value = vet[i];
            j = i;
            while (j > h-1 && value <= vet[j - h]) {
                vet[j] = vet[j - h];
                j = j - h;
            }
            vet[j] = value;
        }
        atualize h;
    }
}
```

Complexidade?

- A complexidade do algoritmo não é conhecida
- A fórmula depende de problemas difíceis
- Como é feito o incremento e escolha do ***h***?
- Existe um ***h*** melhor que os demais?

Como avaliar empiricamente?

- Vamos avaliar a execução do shellsort para diferentes valores de h e regras de atualização
- Vamos comparar com um método de ordenação eficiente.
- O algoritmo depende do vetor de entrada
- Vamos executar o algoritmo para vetores ***sorteados***, representando vetores ***comuns***
- Para isso vamos precisar gerar números aleatórios

Gerar Números Aleatórios

- Em C e C++ podemos gerar número aleatórios com a biblioteca stdlib.h
- A função rand() gera um número aleatório entre 0 e MAX_RAND
- A função rand() precisa ser “reiniciada” para gerar valores diferentes em cada execução
- Para reiniciar a função rand(), devemos usar uma semente diferente. Podemos fazer isso com a função srand()
- Para obter uma semente que dependa da execução podemos fazer: srand(time(NULL))

Gerar Vetores Aleatórios

```c
#include <stdio.h>
#include <stdlib.h>
#include<time.h>
int main(void){
    int i;
    int vet[10];
    printf("Gerando 10 valores aleatorios:\n\n");
    srand(time(NULL));
    for (i=0; i < 10; i++){
        vet[i] = rand() % 100;
    }
    return 0;
}
```

Roteiro da Prática de Hoje

1. Planejar uma bateria de testes com diferentes valores de tamanho para os vetores a serem gerados pelo programa, assim como uma forma de analisar a performance do método de maneira significativa para a sua escolha de h's.
2. Implementar o heapsort (ordenação por heap) e o shellsort com a sua estratégia.
3. Implementar um gerador de vetores aleatórios
4. Realizar os testes descritos no ponto 1.
5. Apresentar um relatório em PDF com os resultados obtidos.