Nome: Guilherme Lirioberto da Silva Alves Disciplina: Estrutura de Dados Data: 24/02/2025

Pesquisa sobre Algoritmos de Ordenação e Busca

Abaixo está uma pesquisa detalhada sobre os algoritmos de ordenação e busca, incluindo definição, algoritmo, vantagens, desvantagens e referências bibliográficas.

---

1. Selection Sort

- **Definição**: Seleciona o menor elemento da lista e o troca com a primeira posição, repetindo o processo para o subvetor restante.
- **Algoritmo**:

```
for (i = 0; i < n-1; i++) {
    min = i;
    for (j = i+1; j < n; j++) {
        if (arr[j] < arr[min]) min = j;
    }
    troca(&arr[min], &arr[i]);
}
```

- **Vantagens**: Simples de implementar e usa pouca memória auxiliar.
- **Desvantagens**: Complexidade O(n2)$O(n2)$ em todos os casos, ineficiente para grandes conjuntos.
- **Referência**: Cormen, T. H. et al. *Introduction to Algorithms*. MIT Press, 2009.

---

2. Insertion Sort

- **Definição**: Constrói a lista ordenada inserindo um elemento por vez na posição correta.
- **Algoritmo**:

```
for (i = 1; i < n; i++) {
    chave = arr[i];
    j = i-1;
    while (j >= 0 && arr[j] > chave) {
        arr[j+1] = arr[j];
```

```
        j--;
    }
    arr[j+1] = chave;
}
```

- **Vantagens**: Eficiente para listas pequenas ou parcialmente ordenadas.
- **Desvantagens**: Complexidade O(n2)$O(n2)$ no pior caso.
- **Referência**: Sedgewick, R. *Algorithms in C*. Addison-Wesley, 1997.

---

3. Bubble Sort

- **Definição**: Compara pares adjacentes e troca-os se estiverem fora de ordem, repetindo até a lista estar ordenada.
- **Algoritmo**:

```
for (i = 0; i < n-1; i++) {
    for (j = 0; j < n-i-1; j++) {
        if (arr[j] > arr[j+1]) troca(&arr[j], &arr[j+1]);
    }
}
```

- **Vantagens**: Simples de implementar.
- **Desvantagens**: Complexidade O(n2)$O(n2)$, ineficiente para grandes conjuntos.
- **Referência**: Knuth, D. E. *The Art of Computer Programming*. Volume 3, 1997.

---

4. Heap Sort

- **Definição**: Constrói uma heap a partir dos dados e extrai o maior/menor elemento repetidamente.
- **Vantagens**: Complexidade O(nlog⁡n)$O(n\log n)$ garantida.
- **Desvantagens**: Não estável e usa memória auxiliar para a heap.
- **Referência**: Cormen, T. H. et al. *Introduction to Algorithms*. MIT Press, 2009.

---

5. Radix Sort

- **Definição**: Ordena dígito por dígito, do menos significativo ao mais significativo.
- **Vantagens**: Eficiente para números inteiros (O(nk)$O(nk)$, onde kk é o número de dígitos).

- **Desvantagens**: Não funciona bem com números negativos ou decimais sem adaptação.
- **Referência**: Skiena, S. S. *The Algorithm Design Manual*. Springer, 2008.

---

6. Quick Sort

- **Definição**: Divide o vetor usando um pivô e ordena recursivamente as partições.
- **Algoritmo**:

```
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi-1);
        quickSort(arr, pi+1, high);
    }
}
```

- **Vantagens**: Média O(nlogn)$O(n \log n)$, rápido na prática.
- **Desvantagens**: Pior caso O(n2)$O(n^2)$ se o pivô for mal escolhido.
- **Referência**: Sedgewick, R. *Algorithms in C*. Addison-Wesley, 1997.

---

7. Merge Sort

- **Definição**: Divide o vetor em metades, ordena recursivamente e combina (merge).
- **Vantagens**: Estável e complexidade O(nlogn)$O(n \log n)$ garantida.
- **Desvantagens**: Usa memória auxiliar para o merge.
- **Referência**: Knuth, D. E. *The Art of Computer Programming*. Volume 3, 1997.

---

8. Busca Sequencial

- **Definição**: Percorre a lista elemento a elemento até encontrar o valor.
- **Complexidade**: O(n)$O(n)$.
- **Uso**: Listas não ordenadas.
- **Referência**: Cormen, T. H. et al. *Introduction to Algorithms*. MIT Press, 2009.

---

9. Busca Binária

- **Definição**: Divide a lista pela metade repetidamente, comparando com o elemento central.
- **Complexidade**: O(log⁡n)$O(\log n)$.
- **Uso**: Requer lista ordenada.
- **Referência**: Cormen, T. H. et al. *Introduction to Algorithms*. MIT Press, 2009.

Referências Bibliográficas

1. Cormen, T. H. et al. *Introduction to Algorithms*. MIT Press, 2009.
2. Sedgewick, R. *Algorithms in C*. Addison-Wesley, 1997.
3. Knuth, D. E. *The Art of Computer Programming*. Volume 3, 1997.
4. Skiena, S. S. *The Algorithm Design Manual*. Springer, 2008.