Aulas de Estruturas de Dados e algoritmos 2

cotrim149

August 29, 2014

1 Busca Sequencial

1. Complexidade média(Tempo de demora para resposta):

$$\frac{n}{2} \tag{1}$$

2. O(n)
3. Métodos para otimização
   - Sentinela: Consiste em adicionar um elemento de valor x no final do vetor.
4. Alternativa: Lista encadeada
5. Aumento de eficiência
   - Método mover para frente: Sempre que uma pesquisa obter êxito, o registro recuperado é colocado no ínicio da lista. **Desvantagem:** Qualquer informação fica privelegiada
   - Método da transposição: Um registro recuperado com sucesso é trocado imediatamento com o elemento anterior (swap é O(1), não importando a quantidade de elementos). **Desvantagem:** Cancelamento da otimização, (swap alternados entre mesmos elementos)
6. Tabela Ordenada
   - Complexidade: O(n/2). **Pior caso:** Complexidade: O(n)
   - Dificuldade: Manter tabela ordenada e a ordenação em si
7. Tabela indexada
   - Utilização de tabela auxiliar como tabela de índices
   - Cada elemento na tabela de índices contém uma chave (kindex) e um indicador do registro no arquivo que corresponde a kindex
8. Vantagens e desvantagens na busca sequêncial

- Vantagens: Os ítens poderão ser examinados sem serem acessados, o tempo de busca diminui consideravelmente
- Desvantagens: Tabela tenha que estar ordenada, demanda mais espaço.

9. Remoção
   - Remova-se o elemento e rearranja-se a tabela
   - Indicar que o local está vazio, e futuramente é inserido outro elemento no índice
10. Inserção
    - Se houver espaço vago, rearranjam-se os elementos localmente, caso não haja espaço, toda a tabela deve ser rearranjada

2 Busca Binária

- O(log n); Cada comparação reduz o número de possíveis candidatos por um fator de 2.
- Pode ser usada como organização de tabela sequencial indexada
- Desvantagem: Precisa de índices, não funciona em uma lista encadeada ou duplamente encadeada

3 Busca por interpolação

- As chaves precisam estar uniformemente distribuidas

$$meio = inf + (sup - inf) * \frac{(x - A[inf])}{(A[sup] - A[inf])} \tag{2}$$

- O(log(log(n))) se as chaves estiverem uniformemente distribuida
- Se chaves não estiverem uniformemente distribuidas, a busca por interpolação pode ser tão ruim quanto uma busca sequencial
- Desvantagem: Em situações práticas, as chaves tendem a se aglomerar em torno de determinados valores e não são uniformemente distribuidas

4 Busca em árvore

1. pré-ordem (sempre a esquerda): [8,3,1,6,4,7,10,14,13]
2. in-ordem (sempra em baixo): [1,3,4,6,7,8,10,13,14]
3. pós-ordem (sempra a direita) : [1,4,7,6,3,13,14,10,8]

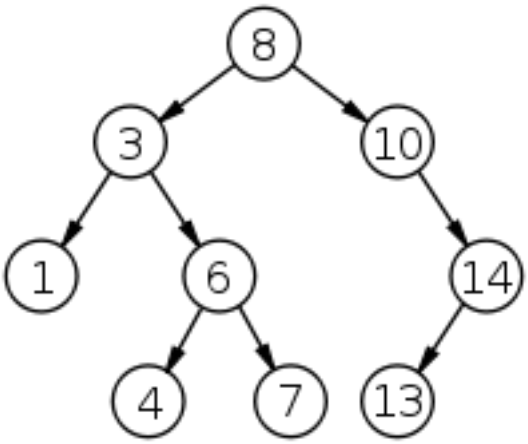

Figure 1: Imagine uma linha passando sempre a esquerda de cada valor começando pela raiz

5 Selection Sort

- Chamado de algoritmo natural
- Se baseia em passar o menor valor do vetor para a primeira posição
- complexidade média = (n-1)*(n/2)
- O(n2)

6 Insertion Sort

- Chamado de algoritmo natural
- Não existe swap para fora do vetor, sempre acrescente o "procurado" entre os valores
-

$$complexidadeMedia = (n) * (\frac{n}{4}) \tag{3}$$

$$Complexidade = O(n^2) \tag{4}$$

7 Bubble sort

$$Complexidade = O(n^2) \tag{5}$$

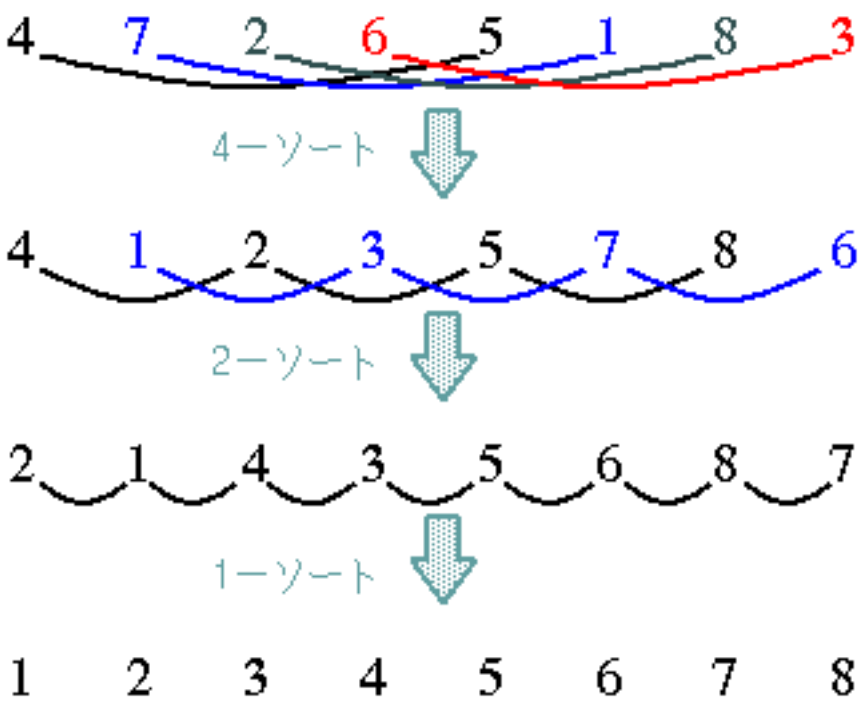

Figure 2: **Shell Sort** Linha 1: Gap de n=4 com 4 subvetores; Linha 2: Gap de n=2 com 2 subvetores; Linha 3: Gap de n=1 com 1 subvetor, Linha 4: Ordenação completa

8 Shell sort

- O mais eficiente algoritmo de ordenação dentre os de complexidade quadrática
- "Gaps" de n/2, sempre fazer o arrendondamento para baixo.
- Vizinho é igual a um "gap" de distância
- Refinamento do insertion Sort
- subvetores dentro do vetor(ideia!), subvetor com tamanho n/"gap"

$$Complexidade = O(n^2) \tag{6}$$

9 Bucket Sort

- "Dividir e conquistar"

- Divisão em "baldes" por faixas de valores,
- pode ser usado qualquer algortimo dentro de cada balde

10 Quick Sort

- "Dividir e conquistar"
- Escolhe-se um valor pivô e move-se todos os valores menores para a esquerda e os maiores para a direita
- Ordena-se recursivamente os valores menores e maiores
- Algoritmo instável: Não garante que elementos iguais não invertam, não se preocupa com a ordem dos elementos
- O(log(n))
- Desvantagem: A escolha de um mau pivô seguidamente podem tornar o algoritmo O(n2)
- Encontrar o mediano é O(n), o que resulta O(n log(n))
- Algoritmo recursivo

11 Merge Sort

- Algoritmo é recursivo, mas pode ser iterativo, existem mudanças de funcionamento.
- O(n log(n))

12 Heap Sort

- Prioriza a busca de valores
- o Pai sempre é maior ou igual que seus filhos(se maior na raiz), ou, o pai sempre é menor que seus filhos(se menor na raiz)
- Último nível sempre tem um buraco a direita. Isso garante que não haverá outros elementos dentro do vetor.
- Vetor nomeado de 1 a n, não de 0 a n-1
- Para encontrar o pai i/2
- Filhos a esquerda 2i
- Filhos a direita 2i + 1

12.1 Inserção

1. O(log(n))
2. Para se fazer um novo nível, sempre tenha que ter o nível imediatamente anterior completo.

12.2 Heapify

- Manipula a estrutura para ela se tornar um heap
- Apartir de uma posição vai consertando para baixo

12.3 Build a Heap

- Transforma qualquer vetor em Heap
- Metade dos valores não são utilizados em Heapfy, pois são folhas.
- Sempre faça Heapfy no pai das folhas.
- Heapdy resolve tudo aqui!
- Complexidade: somatorio : (n/2 * 0) + (n/4 * 1) + (n/8 * 3) + ...

12.4 HeapSort

- Feito localmente, não é recursivo
-

12.5 Extra-Max

13 Radix Sort

13.1 Count sort

- Vetor 'A' - Vetor analisado com valores aleatórios
- Vetor 'C' - Vetor com tamanho do maior valor encontrado no vetor 'A'
- Faça um vetor 'C' com o valor da quantidade de cada numero do vetor 'A', cada indice do vetor 'C' refere-se a quantidade daquele indice no vetor 'A'
- Faça uma soma de uma celula com a imediatamente anterior do vetor 'C'
- A partir do vetor 'A' faça a análise no vetor 'C' de qual posição está o valor procurado.Diminui em 1 o valor que foi acessado
- O(n+k)