Aluno: Christian Jonas Oliveira

Estrutura de Dados II
Avaliação 1

Observação 1: Caso seja identificado cola, todos os envolvidos receberão nota zero.

Questão 1: (2,5 pontos) Considerando os métodos: *selection sort*, *selection sort* e *bubble sort*, qual dos métodos é mais rápido para ordenar um array cujos elementos já estão ordenados (do menor para o maior)? Justifique apresentando a análise de complexidade do caso para cada algoritmo.

Primeiramente, analisarei o selection sort. Para esse algoritmo, com o objetivo de encontrar o menor, cada elemento será comparado com todos os outros. Como existem dois laços nesse algoritmo, um dentro do outro, e pela maneira que o algoritmo se dá, percebemos que independente da ordem das chaves, o algoritmo terá complexidade $O(n^2)$, uma vez que o a cada iteração do laço mais externo, o laço mais interno será executado "n" vezes.

Agora, analisarei o insertion sort. Nesse algoritmo para cada iteração feita pelo laço mais externo, neste caso, nunca "entrará" no laço mais interno, uma vez que o vetor já estará ordenado. Portanto, neste caso, a complexidade do algoritmo é O(n).

Por último analisarei o bubble sort. Nesse algoritmo, há duas estruturas de repetição. Para nosso caso, onde o array já estará ordenado, temos complexidade O(n), no entando, como há muitas comparações necessárias, esse algoritmo não se sai bem para um número alto de entradas, mostrando que varia muito no que se refere ao tempo de execução.

Com isso, podemos concluir que o melhor algoritmo para este caso é o Insertion Sort.

Questão 2: (2,0 pontos) O método *Selection sort* é estável? Justifique.

Não é estável, uma vez que o algoritmo faz sucessivas trocas ele não preserva a precedência dos termos repetidos.

Exemplo:

[1, 4', 4, 2]	1ª it.
[1, 2, 4, 4']	última iteração

Percebemos que o 4' vem depois de 4 ao findar o processo do algorítmo. Portanto não preserva a precedência.

Questão 3: (2,5 pontos) Verdadeiro ou Falso: O tempo de execução do algoritmo *Radix sort* não depende da ordem das chaves no *array* de entrada. Justifique sua resposta.

Falso, pois o mesmo, utiliza um outro algoritmo para ordenar os dígitos. Se o algoritmo escolhido para ordenação não for um algoritmo eficiente, o impacto no tempo de execução será claro. Um exemplo é pegar a diferença entre um Radix implementado com um Bubble Sort e um Radix implementado com um Insertion Sort, onde o algoritmo implementado com o Insertion Sort, será bem mais eficiente.

Questão 4: (3,0 pontos) Suponha que temos apenas 3 fitas disponíveis para ordenar um conjunto de registros e os seguintes blocos foram obtidos pelo método de seleção por substituição: [*A, O, P, Q, T*], [*I, S*], [*A, H, N*], [*D, E, P, R*], [*G, L, N*], [*E, G, X*], [*A, M, P*], [*E, F*]. Qual é distribuição dos blocos nas fitas para iniciar o processo de intercalação polifásica? Apresente o cálculo realizado para descobrir tal distribuição. Note que o processo deve estar restrito à utilização de 3 fitas.

Analisando a situação, montei a tabela começando do final, como explicado em aula.

F 1	F 2	F 3	TOTAL
0	5	3	8
3	2	0	5
1	0	2	3
0	1	1	2
1	0	0	1

Vale ressaltar que nesse método o objetivo é sempre ter uma fita vazia, afim de armazenar o resultado da intercalação das outras fitas.

F1 – sem elementos inicialmente, pois escolhi ela para armazenar minha ordenação final;
F2 - [*D, E, P, R*] [*I, S*] [*E, F*] [*E, G, X*] [*A, H, N*] .
F3 - [*A, O, P, Q, T*] [*G, L, N*] [*A, M, P*].