Gustavo Henrique Storti Salibi / RA: 174135 / Turma: F

Giovanni Bertão / RA: 173325 / Turma: E

Exercício em Dupla 01

Q1.

a) *Selection-Sort:* O(n²);
Insertion-Sort: O(n²);
Merge-Sort: O(n*log n);
Quick-Sort: O(n²).
(Foram consideradas as ordens de grandeza dos piores casos).

b) *Selection-Sort:* 15 comparações e 1 troca (OBS: seguindo o algoritmo visto em sala, o procedimento de troca é realizado mesmo que seja para a mesma posição. Assim, teríamos 6 trocas);
Insertion-Sort: 15 comparações e 1 troca.
Merge-Sort: 9 comparações.(OBS: cada recursão executou 3 comparações, incluindo a primeira, isso justifica as 9 comparações. Contudo, após a 6ª comparação a sequência já estava organizada).
Quick-Sort: 15 comparações e 5 trocas. (OBS: esses valores foram obtidos seguindo os algoritmos. Contudo, em 14 comparações e 4 trocas a sequência já estava ordenada).

Sim, os resultados são compatíveis. No exercício anterior, foram considerados os piores casos, assim, os resultados não tiveram exatamente a mesma complexidade do exercício, mas respeitaram a ordem de eficiência dos algoritmos exemplificados, onde o Merge-Sort se saiu melhor. Além disso, no exemplo utilizado tínhamos um teste de somente 6 elementos, isso justifica a proximidade dos resultados, caso a sequência fosse maior, teríamos uma maior diferença. Além disso, existem melhores e piores casos para cada algoritmo.

Q2.

Selection-Sort:
Pior caso: A sequência está ordenada da forma decrescente(já que queremos uma sequência ordenada crescentemente).
Melhor caso: A sequência já está ordenada como queremos.

Insertion-Sort:
Pior caso: Sequência ordenada inversamente.
Melhor caso: Sequência já ordenada.

Merge-Sort:
Pior caso: Seja a sequência dada por: a1 <= a2 <= … <= a6. Se a sequência original estiver organizada no seguinte formato: a1, a3, a5, a2, a4, a6. Teremos o pior caso. Pois, o algoritmo de intercalar realizará o maior número de comparações. Ex: 2 4 6 3 5 7.
Melhor caso: A sequência já está ordenada garante o menor número de comparações ao intercalar.

Quick-Sort:
Pior caso: A sequência já está ordenada. Pois, cada vez que particionar a sequência teremos uma partição de n-1 elementos menores que o pivô e 1 elemento(no caso o pivô da partição) na outra partição. Ex: 2 2 3 4 9 A.
Melhor caso: Cada partição divide a sequência original em exatamente 2 sequências de mesmo tamanho. Ex: 2 3 2 9 A 5.

Q3.

a) Pode-se ignorar as 24 primeiras cartas, já que elas são utilizadas de toda forma. Com isso, começa-se a verificar a partir da 25ª e, caso haja alguma carta fora de ordem (ignorando a sequência de cartas utilizadas no pife-pafe), a verificação pode ser encerrada e quem as utilizou foi o amigo que joga pife-pafe, caso se chegue à última carta e não se tenha carta desordenada, o outro amigo, do truco, as utilizou.

b) Usando a identificação anterior, descobrimos qual dos dois amigos usou o baralho.

Caso seja o jogador de truco:

- Só precisamos organizar as 24 primeiras cartas. Temos um baralho inteiro já organizado e um baralho de 28 cartas organizadas;
- Ordenamos as cartas por naipes, formando assim 4 montes;
- Realizamos qualquer algoritmo de ordenação com cada um dos montes;
- Adicionamos, na posição correta, cada uma das cartas de cada um dos montes ao baralho de 28 cartas seguindo, também, a ordem dos naipes;
- Ao final, teremos organizado um baralho, e, como o outro já estava organizado, teremos dois baralhos organizados.

Caso seja o jogador de pife-pafe:

- Precisamos organizar todas as 104 cartas;
- Separamos todas as cartas por naipes, nos garantindo 4 montes de cartas desordenadas;
- Para cada monte, realizamos um algoritmo de ordenação;
- Percorremos um monte de cada vez removendo as cartas iguais e adicionando elas em um novo monte de naipes iguais de forma a termos 4 pares de montes iguais no final;
- Usando 4 dos 8 montes anteriores, fazemos um único monte, onde não devemos utilizar dois montes de mesmo naipe para formá-lo;
- Com os 4 montes restantes, formamos outro monte igual ao anterior;
- Ao final, teremos dois baralhos ordenados.