Pesquisa e Classificação de Dados
Lista 2 (Busca e Ordenação)

Prof. Ricardo Oliveira

OBS: Apenas o resultado final é apresentado. Realize também o desenvolvimento do exercício!

OBS2: Caso encontre algum erro, por favor avise o professor.

1. (a) $O(B \times N)$
   (b) $O((N + B) \lg N)$
   (c) Não ordenar e utilizar a busca linear
   (d) Ordenar e utilizar busca binária
   (e) Ordenar e utilizar busca binária
2. Implemente o laço interno do *InsertionSort*.
3. O custo de pior caso total será de $O(\lg N + N) = O(N)$, assintoticamente indêntico ao do algoritmo anterior.
4. Melhor Caso = $O(1)$. Pior caso = Caso médio = $O(\lg N)$ (Assim como a busca binária comum).
5. Exercício de implementação
6. $O(N^2)$
7. Exercício de implementação
8. (a) Conte o número de trocas adjacentres realizadas
   (b) Conte o número de trocas adjacentres realizadas
   (c) No laço interno, ao descolar o menor elemento, compute a distância de sua posição atual à sua posição correta (por quê?)
   (d) Uma descrição do algoritmo pode ser encontrada em `https://www.geeksforgeeks.org/counting-inversions/`.
9. O algoritmo continua $\Theta(N \lg N)$ (Não há "aprimoramento" de fato).
10.
```
Para t = 1,2,4,8,16,...
    Para i=0, 2t, 4t, 6t, ...
        merge(v[i..i+t-1], v[i+t..i+2t-1])
```
11. Gere o vetor ordenado.
12. Embaralhe o vetor aleatoriamente antes de ordená-lo (pois o caso médio do *QuickSort* é $O(N \lg N)$).
13. (a) $[5, 4, 1, 2, 3]$
    (b) $[4, 3, 1, 2, 5]$
    $[3, 2, 1, 4, 5]$
    $[2, 1, 3, 4, 5]$
    $[1, 2, 3, 4, 5]$
14. O *CountingSort* não é *in-place* e requer $\Theta(M)$ espaço extra. Um vetor de 2^{32} posições inteiras ocupa 16 Gb de memória, cuja alocação é inviável.
15. $O(\frac{N}{2} \lg \frac{N}{2})$ (ainda $O(N \lg N)$).

16. $O(\frac{N}{B} \lg \frac{N}{B})$.
17.
```
321     321     321
427     427     427
578 -> 458 -> 458
458     578     578
```
18. A estabilidade garante a ordenação lexicográfica do vetor. No exercício anterior, um método instável pode, por exemplo, trocar a ordem de 427 e 458 no último passo do algoritmo.
19. São (ou podem ser implementados como) estáveis: *Bubble*, *Insertion*, *Selection*, *Merge*, *Bucket** e *Radix*. São *in-place*: *Bubble*, *Insertion*, *Selection*, *Quick*, *Heap*, e *Radix**. (*dependendo do método usado como subrotina, incluindo a possibilidade dos quadráticos).
20. Exercício de implementação, mas note que, devido ao alto limite para N, algoritmos quadráticos não devem ser aceitos. Implemente um algoritmo $O(N \lg N)$.