MATA52 - Exercícios da Semana 04

- Grupo: Paladio
- Autores:
  - Monique Silva (Responsável)
    - Disponibilizei a lista de exercícios da semana 4 para os membros terem acesso, no canal do grupo no discord, marquei para reunirmos na quinta-feira (31/02) com o objetivo de resolver as questões da lista. Todos os membros estiveram presente no dia. Em relação a lista, respondi a quarta questão, com base no aprendizado sobre tempo de execução, complexidade e ordenação bucket, além de escutar e debater com meus colegas sobre a resolução de suas questões.
  - Lucas dos Santos Lima
    - Respondi a segunda questão e expliquei o algoritmo e a forma que eu havia colocado na questão para os colegas de grupo, a partir de pesquisas na internet, da aula e do livro proposto para a disciplina.
  - Bruno de Lucas Santos Barbosa
    - Respondi a primeira questão explicando meu entendimento sobre a árvore de decisões além de contribuir com o Lucas para o entendimento do algoritmo da segunda questão
  - Elis Marcela de Souza Alcantara
    - Resolvi a terceira questão e auxiliei na explicação do Radix MSD e o Radix LSD.

Instruções (não apagar)

1. **Responsável**: Após copiar este notebook, altere o nome do notebook/documentação incluindo o nome do seu grupo. Por exemplo, se você é do grupo Ouro, altere o nome do notebook para "MATA53-Semana02-Ouro.ipynb"
2. **Responsável**: Compartilhe este documento com todos os membros do grupo (para participarem da elaboração deste documento). É importante que o notebook utilizado seja o mesmo compartilhado para que os registros de participação e colaboração fiquem salvos no histórico. Sugira uma divisão justa e defina um prazo aceitável para a inserção das soluções no Colab.

3. **Responsável**: Ao concluir a atividade, compartilhe o notebook com januario.ufba@gmail.com (dando permisão para edição) e deixando o aviso de notificação marcado, para que eu receba o seu e-mail. Identificar o nome do grupo na mensagem de compartilhamento.
4. **Cada membro**: Incluia o *seu próprio nome completo* na lista de autores que auxiliaram na elaboração deste notebook. Relate brevemente a sua contribuição na solução desta lista. O responsável aparece como sendo o(a) primeiro(a) autor(a).
5. **Cada membro**: Utilize os recursos de blocos de texto e de código para adicionar as suas respostas, sem alterar os blocos de texto e código existente. Não economize, esses blocos são de graça.

Exercícios

1. Qual seria a menor profundidade possível de uma folha em uma árvore de decisão para um processo de ordenação por comparação de n elementos?

Resposta 1:

A altura mínima de uma árvore de decisões é de $(n \log n)$, já a menor profundidade possível seria de $n - 1$.

$n - 1$ indica que o array já está ordenado, então é a profundidade que representa o melhor caso, pois realizamos o menor número de comparações para determinar uma ordem de classificação adequada.

2. Apresente o passo a passo do algoritmo de ordenação por contagem para a sequência $A = (6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2)$.

Resposta 2:

Temos a seguinte lista: A = (6,0,2,0,1,3,4,6,1,3,2) O maior número seria o 6, assim criamos uma lista auxiliar com 6 + 1 espaços preenchidos com valor 0: Auxiliar = (0,0,0,0,0,0,0) Nesse array iremos

contar a quantidade de vezes que um número aparece de acordo com seu índice. Além disso precisaremos de um array de soma cumulativa, que irá guardar as posições de cada alemento na futura lista organizada. Assim, começamos a percorrer a lista principal e os colocaremos nas seguintes posições: A[0] = 6, somamos 1 na posição 6 da lista auxiliar, logo temos Auxiliar[6]=1

A[1] = 0, somamos 1 na posição 0 da lista auxiliar, logo temos Auxiliar[0]=1

A[2] = 2, somamos 1 na posição 2 da lista auxiliar, logo temos Auxiliar[2]=1

A[3] = 0, somamos 1 na posição 0 da lista auxiliar, logo temos Auxiliar[0]=2

A[4] = 1, somamos 1 na posição 1 da lista auxiliar, logo temos Auxiliar[1]=1

A[5] = 3, somamos 1 na posição 3 da lista auxiliar, logo temos Auxiliar[3]=1

A[6] = 4, somamos 1 na posição 4 da lista auxiliar, logo temos Auxiliar[4]=1

A[7] = 6, somamos 1 na posição 6 da lista auxiliar, logo temos Auxiliar[6]=2

A[8] = 1, somamos 1 na posição 1 da lista auxiliar, logo temos Auxiliar[1]=2

A[9] = 3, somamos 1 na posição 3 da lista auxiliar, logo temos Auxiliar[3]=2

A[10] = 2, somamos 1 na posição 2 da lista auxiliar, logo temos Auxiliar[2]=2

Agora teremos Auxiliar = (2, 2, 2, 2, 1, 0, 2), relembrando que temos Auxiliar[5] = 0 pois o número 5 não aparece em A.

Agora teremos uma lista de soma acumulativa que chamaremos de Acumulativa e irá armazenar as posições (índices) que cada número da lista original preencherá na lista ordenada, ou ao menos nos dizerá a última posição em que ele está presente, e podemos deduzir que entre ele e a última posição do número anterior, teremos somente cópias dele.

Assim, podemos seguir o raciocínio que Acumulativa[i] = ∑{t=0]}^{i}Auxiliar[t]. Então teremos:

Acumulativa[0] = Auxiliar[0]

Acumulativa[1] = Auxiliar[0] + Auxiliar[1]

Acumulativa[2] = Auxiliar[0] + Auxiliar[1] + Auxiliar[2]

Assim sucessivamente para cada elemento da lista Acumulativa, por fim teremos: Acumulativa = (2, 4, 6, 8, 9, 9, 11)

Agora a partir da lista acumulativa seguiremos o seguinte raciocínio: Cada índice dessa lista indicará o valor que colocaremos na lista ordenada e o número presente nesse índice reduzido de 1 será a última posição em que colocaremos aquele número*.

- reduziremos 1 pois no começo da explicação somamos 1 no maior valor presente na lista A e fizemos desse número o número de posições da lista Auxiliar, para que o maior índice da lista Auxiliar igualasse o maior valor na lista A.

Então temos a lista ordenada que chamaremos de Final = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Aplicando a lógica supracitada temos:

Acumulativa[0] = 2, na posição 2-1 de Final colocaremos 0: Final[1] = 0

Acumulativa[1] = 4, na posição 4-1 de Final colocaremos 1: Final[3] = 1

Acumulativa[2] = 6, na posição 6-1 de Final colocaremos 2: Final[5] = 2

Acumulativa[3] = 8, na posição 8-1 de Final colocaremos 3: Final[7] = 3

Acumulativa[4] = 9, na posição 9-1 de Final colocaremos 4: Final[8] = 4

Acumulativa[5] = 9, na posição 9-1 de Final colocaremos 5: Final[8] = 5, porém nesse caso não colocaremos nada pois sabemos dois fatos:

1. O número 5 não aparece na lista original
2. Essa posição já está ocupada

Acumulativa[6] = 11, na posição 11-1 de Final colocaremos 6: Final[10] = 6

Então a lista Final que temos até agora é: Final = (0, 0(preenchido com 0), 0, 1, 0, 2, 0, 3, 4, 0, 6)

Porém como já dito, a primeira posição que inserimos um número na lista final é o ÚLTIMO lugar onde ele aparece, dessa forma é razoável imaginar que havendo espaço antes desse número sem ser preenchido, o preencheremos com o número seguinte.

Então temos Final = (0, 0, 1, 1, 2, 2, 3, 3, 4, 6, 6).

De maneira resumida:

1. Encontramos o maior número no array A
2. Num array auxiliar colocamos a quantidade de cada elemento de A em seu índice correspondente
3. Num array acumulativo em cada índice colocamos a soma dos elementos do índice 0 até o índice correspondente no array auxiliar
4. Num array ordenado final (poderia ser até mesmo no array original) substituímos cada índice do array Acumulativo na posição n - 1 do array ordenado final, sendo n o valor presente no índice do array Acumulativo.
5. Por fim garantimos que qualquer posição não preenchida do array ordenado final fosse preenchido com o número maior seguinte, ou logicamente a cada substituição na posição n - 1, reduzimos 1 de n.

3. Apresente o passo a passo do algoritmo de ordenação radix para as palavras: COW, DOG, SEA, RUG, ROW, MOB,

BOX, TAB, BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX.

Resposta 3:

Utilizando o LSD radix sort que começa a ordenar a partir do fim das strings (LSD = dígito menos significativo)

Conjunto
$S = COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB, BAR, EAR, TAR, DIG, BI$
Todas as strings do conjunto S tem tamanho 3.

1ª Coluna: comparação entre o dígito menos significativo __X
2ª Coluna: ordem alfabética a partir do dígito menos significativo e comparação entre o dígito do meio _X_
3ª Coluna: ordem alfabética a partir do dígito do meio e comparação entre o dígito do começo X__
4ª Coluna: ordem alfabética finalizada

CO**W** ➡ S**E**A ➡ **T**AB ➡ BAR
DO**G** ➡ T**E**A ➡ **B**AR ➡ BIG
SE**A** ➡ M**O**B ➡ **E**AR ➡ BOX
RU**G** ➡ T**A**B ➡ **T**AR ➡ COW
RO**W** ➡ D**O**G ➡ **S**EA ➡ DIG
MO**B** ➡ R**U**G ➡ **T**EA ➡ DOG
BO**X** ➡ D**I**G ➡ **D**IG ➡ EAR
TA**B** ➡ B**I**G ➡ **B**IG ➡ FOX
BA**R** ➡ B**A**R ➡ **M**OB ➡ MOB
EA**R** ➡ E**A**R ➡ **D**OG ➡ NOW
TA**R** ➡ T**A**R ➡ **C**OW ➡ ROW
DI**G** ➡ C**O**W ➡ **R**OW ➡ RUG
BI**G** ➡ R**O**W ➡ **N**OW ➡ SEA
TE**A** ➡ N**O**W ➡ **B**OX ➡ TAB
NO**W** ➡ B**O**X ➡ **F**OX ➡ TAR
FO**X** ➡ F**O**X ➡ **R**UG ➡ TEA

4. Explique porque o pior caso da ordenação por bucket pode chegar a $\theta(n^2)$. Quais alterações devem ser feitas pra a

obtenção de um pior caso sendo $O(n \lg n)$?

Resposta 4

O tempo de execução do pior caso para o algoritmo bucket-sort ocorre quando a entrada dos valores não é uniformemente distribuída. Se, por exemplo, toda a entrada terminar no primeiro bucket, então na fase de ordenação por inserção ele precisa ordenar toda a entrada, o que leva $O(n^2)$.

Para obtermos o tempo de execução do pior caso $O(n \lg n)$ seria necessário alterar para usar um algoritmo de tempo do pior caso, como o merge sort, em vez de insert sort ao ordenar os buckets.