1290 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA DEPARTAMENTO DE ENGENHARIA INFORMÁTICA	**Trabalho no 4 v2 – Palavras mais procuradas** **Algoritmos e Estruturas de Dados** **2019/2020 – 2º Semestre** Upload: (link a disponibilizar no infoestudante) Data Limite: até 15mn depois do fim da respetiva 4ª aula pratica

O RELATÓRIO E LISTAGEM DO CÓDIGO DESENVOLVIDO DEVEM SER SUBMETIDOS NUM ÚNICO DOCUMENTO PDF

Nome: Rodrigo Fernando Henriques Sobral nº: 2018298209 PL: 2

Nº de horas de trabalho: Aulas Práticas de Laboratório: 8H Fora de Sala de Aula: 5H

CLASSIFICAÇÃO: (A Preencher pelo Docente)

Análise Empírica de Complexidade

- Faça o download dos 4 textos. Preencha a Tabela 1 com os valores para cada texto. Use o tempo médio de 20 execuções. Considere essa caracterização na análise qualitativa mais adiante.
- Preencha a Tabela 2 com informação sobre as estruturas de dados e algoritmos que escolheu.
- Preencha a Tabela 3 com os tempos de ordenamento para cada algoritmo.

#	Operação	Texto A	Texto B	Texto C	Texto D
1	Carregamento (tempo em: s)	0 .07	130 .3	165 .1	153 .05
2	Núm. palavras distintas	75	1 354	1 354	1 354
3	Núm. utilizadores distintos	2	632	632	632
4	Núm. Pares (palavra,utilizador) distintos	150	4 159	4 159	4 159
5	Núm. Total de palavras	156	140 000	140 000	140 000
6	Núm. Total de utilizadores	156	140 000	140 000	140 000
7	Faz algum ordenamento? Caracterize.	Alfabeticamente crescente e aumento progressivo do número de caracteres.	Não.	Repetição massiva e consecutiva e alfabeticamente crescente.	Curta repetição consecutiva e alfabeticamente crescente.

#	Tarefa	Estrutura de dados	Algoritmo de ordenamento

8	A1	Arrays	Merge Sort
9	A2	Arrays	Radix Sort + Counting Sort

#	Tempos de Ordenamento em [µs]	A1 PESQ_GLOBAL	A1 PESQ_UTILIZADORES	A2 PESQ_GLOBAL	A2 PESQ_UTILIZADORES
10	Texto A	347 .05	99 .8	299 .4	249 .1
11	Texto B	7 608 .6	6 541 .0	45 599 .85	28 327 .85
12	Texto C	10 419 .6	8 538 .4	53 265 .4	30 862 .25
13	Texto D	7 843 .85	6 254 .05	51 695 .5	30 853 .65

Reflexão sucinta sobre os resultados obtidos

(Formato de referência: Arial 10pt; texto para além do número de linhas não é considerado e desvaloriza o relatório)

1. Comente como as estruturas de armazenamento definidas na Tabela 2 influenciaram os tempos de armazenamento e processamento.

Se utilizássemos outra estrutura de dados, os tempos de armazenamento não seriam melhores (devido a rotações ou acessos mais demorados a memoria). Porém, no requisito "processamento", uma árvore binária ficaria em vantagem já que os dados estariam *a priori* quase, ou, completamente ordenados.

2. Analise os resultados dos algoritmos para as tarefas A1 e A2. Os resultados foram os esperados? O input mostrou-se relevante? Justifique. (8 linhas)

O algoritmo A1 (Merge Sort) possui complexidade O(n*log(n)), e os dois algoritmos A2, Radix e Counting Sort, complexidades temporais de O((n+b)*logb(k)) e O(n+k), respetivamente. Esta informação vai ao encontro dos dados levantados, já que a comparação de bases mostrou uma maior eficácia em quantidades massivas de informação, como é o caso. Portanto sim, os resultados estão dentro dos esperados. O input também é relevante. No caso A1, quanto maior for o input, mais bifurcações e unificações serão necessárias, o que causa um aumento temporal significativo. No caso A2, quanto mais dígitos o input tiver, mais ciclos e comparações precisam de ser feitas (Radix), e quanto maior o próprio input, mais demorado será o preenchimento da estrutura auxiliar (Counting).

3. Compare as duas formas de ordenamento: por registo vs. por endereço. Quando será apropriada a escolha de cada uma? (6 linhas)

Na utilização dum ordenamento por endereço, não há efetivamente a troca de elementos, apenas a alteração de endereços para os quais os ponteiros apontam. Por outro lado, no ordenamento por registo há um movimento de elementos na memória, o que consequentemente exige um maior tempo de processamento. Posto isto, concluímos que em quantidades massivas de informação, como é o caso, ficaria mais em conta a utilização dum ordenamento por endereço.

Algoritmo A1

```python
from datetime import datetime       You, a day ago • 3 question lefting

def receiveCommands():
    # [ word , [all_searchers] , [distinct_searchers] ]
    database=[]
    results=""
    while True:
        command=input().strip("\n")

        if command=="PALAVRAS":
            startTime= datetime.now()
            totWords=0
            aux=input().strip("\n")
            while (aux!="FIM."):
                totWords+=1
                aux=aux.split(" ")
                index= isIn(database, aux[0])
                if index== -1: database.append([aux[0], [aux[1]], [aux[1]]])
                elif aux[1] not in database[index][1]:
                    database[index][1].append(aux[1])
                    database[index][2].append(aux[1])
                else: database[index][1].append(aux[1])
                aux=input().strip("\n")
            loadingTextTime= (datetime.now()-startTime).seconds
            results+= "GUARDADAS\n"
        elif command=="PESQ_GLOBAL":
            startTime= datetime.now()
            mergeSort(database, 1)
            sortGlobalTime= (datetime.now()-startTime).microseconds
            i=1
            results+=str(database[0][0])
            while i<len(database) and len(database[0][1])==len(database[i][1]):
                results+=" "+str(database[i][0])
                i+=1
            results+="\n"
        elif command=="PESQ_UTILIZADORES":
            startTime= datetime.now()
            mergeSort(database, 2)
            sortUtilizadoresTime= (datetime.now()-startTime).microseconds
            i=1
            results+=str(database[0][0])
            while i<len(database) and len(database[0][2])==len(database[i][2]):
                results+=" "+str(database[i][0])
                i+=1
            results+="\n"
        elif command=="TCHAU": break
        else: print("Incorrect command.")

    #print(results, end='')
    return [database, loadingTextTime, sortGlobalTime, sortUtilizadoresTime, totWords]

def isIn(database, word):
    for search in database:
        if word.upper()==search[0].upper(): return database.index(search)
    return -1

def mergeSort(database, command):
    if len(database)>1:
        mid = len(database)//2
        left_array, right_array = database[:mid], database[mid:]

        mergeSort(left_array, command)
        mergeSort(right_array, command)

        database.clear()
        while len(left_array) > 0 and len(right_array) > 0:
            if len(left_array[0][command]) >= len(right_array[0][command]):
                database.append(left_array[0])
                left_array.pop(0)
            else:
                database.append(right_array[0])
                right_array.pop(0)

        for i in left_array:
            database.append(i)
        for i in right_array:
            database.append(i)

#==================================

def getDistinctUsers(database):
    tot_users=[]
    for search in database:
        for user in search[2]:
            if user not in tot_users: tot_users.append(user)
    return len(tot_users)
def getWordUserPair(database):
    counter=0
    for search in database: counter+=len(search[2])
    return counter

if __name__ == "__main__":
    [database, loadingTextTime, sortGlobalTime, sortUtilizadoresTime, totWords]= receiveCommands()
    for i in range(19):
        results=receiveCommands()
        loadingTextTime+= results[1]
        sortGlobalTime+= results[2]
        sortUtilizadoresTime+= results[3]
    print("================")
    print("DATA:")
    print("Loading Text Time:", loadingTextTime/20)
    print("Distinct Words:", len(database))
    print("Distinct Users:", getDistinctUsers(database))
    print("Distinct Word-User Pair:", getWordUserPair(database))
    print("Total Words:", totWords)
    print("Total Users:", totWords)

    print("Sorting Global Time:", sortGlobalTime/20)
    print("Sorting Utilizadores Time:", sortUtilizadoresTime/20)
```

Algoritmo A2

```python
from datetime import datetime      You, a day ago • 3 question lefting

def receiveCommands():
    database=[]     # [ word , [all_searchers] , [distinct_searchers] ]
    results=""
    while True:
        command=input().strip("\n")

        if command=="PALAVRAS":
            startTime= datetime.now()
            totWords=0
            aux=input().strip("\n")
            while (aux!="FIM."):
                totWords+=1
                aux=aux.split(" ")
                index= isIn(database, aux[0])
                if index== -1: database.append([aux[0], [aux[1]], [aux[1]]])
                elif aux[1] not in database[index][1]:
                    database[index][1].append(aux[1])
                    database[index][2].append(aux[1])
                else: database[index][1].append(aux[1])
                aux=input().strip("\n")
            loadingTextTime= (datetime.now()-startTime).seconds
            results+= "GUARDADAS\n"
        elif command=="PESQ_GLOBAL":
            startTime= datetime.now()
            database= radixSorting(database, 1)
            sortGlobalTime= (datetime.now()-startTime).microseconds
            results+=str(database[0][0])
            i=1
            while i<len(database) and len(database[0][1])==len(database[i][1]):
                results+=" "+str(database[i][0])
                i+=1
            results+="\n"
        elif command=="PESQ_UTILIZADORES":
            startTime= datetime.now()
            database= radixSorting(database, 2)
            sortUtilizadoresTime= (datetime.now()-startTime).microseconds
            results+=str(database[0][0])
            i=1
            while i<len(database) and len(database[0][2])==len(database[i][2]):
                results+=" "+str(database[i][0])
                i+=1
            results+="\n"
        elif command=="TCHAU": break
        else: print("Incorrect command.")

    #print(results, end='')
    return [database, loadingTextTime, sortGlobalTime, sortUtilizadoresTime, totWords]

def isIn(database, word):
    for search in database:
        if word.upper()==search[0].upper(): return database.index(search)
    return -1

def radixSorting(database, command):
    searchesArray= [str(len(search[command])) for search in database]
    longestDigits= len(str(max(searchesArray, key=len)))
    for num_digits in range(longestDigits):
        digitsArray=[]
        for digits in searchesArray:
            if len(digits)-1-num_digits<0: digitsArray.append(-1)
            else: digitsArray.append(int(digits[len(digits)-1-num_digits]))
        database= countingSort(database, digitsArray)
        searchesArray=[str(len(search[command])) for search in database]
    return database

def countingSort(database, digitsArray):
    smaller=min(digitsArray)
    indexArray= [0 for i in range(max(digitsArray)-smaller+1)]
    sortedArray=[]
    for number in digitsArray: indexArray[number-smaller]+=1
    del smaller, number
    for occurrences in range(len(indexArray)-1, -1, -1):
        while indexArray[occurrences]!=0:      You, a day ago • 3 question
            greater=digitsArray.index(max(digitsArray))
            indexArray[occurrences]-=1
            sortedArray.append(database[greater])
            digitsArray.pop(greater)
            database.pop(greater)
    return sortedArray

def getDistinctUsers(database):
    tot_users=[]
    for search in database:
        for user in search[2]:
            if user not in tot_users: tot_users.append(user)
    return len(tot_users)
def getWordUserPair(database):
    counter=0
    for search in database: counter+=len(search[2])
    return counter

if __name__ == "__main__":
    [database, loadingTextTime, sortGlobalTime, sortUtilizadoresTime, totWords]= receiveCommands()
    for i in range(19):
        results=receiveCommands()
        loadingTextTime+= results[1]
        sortGlobalTime+= results[2]
        sortUtilizadoresTime+= results[3]
    print("================")
    print("DATA:")
    print("Loading Text Time:", loadingTextTime/20)
    print("Distinct Words:", len(database))
    print("Distinct Users:", getDistinctUsers(database))
    print("Distinct Word-User Pair:", getWordUserPair(database))
    print("Total Words:", totWords)
    print("Total Users:")

    print("Sorting Global Time:", sortGlobalTime/20)
    print("Sorting Utilizadores Time:", sortUtilizadoresTime/20)
```

Bom trabalho, os Docentes da Disciplina,

Carlos L Bento e Catarina Silva