FUNDAÇÃO GETÚLIO VARGAS
ESCOLA DE MATEMÁTICA APLICADA
MESTRADO 2015.1
ESTRUTURA DE DADOS E SEUS ALGORITMOS
Prof Alexandre Rademaker

QuickSort: Análise da escolha dos pivôs

GRUPO:
KIZZY TERRA
OTTO TAVARES
VIVIAN TOMÉ

RIO DE JANEIRO
ABRIL DE 2015

1 Implementação do QuickSort

O método quicksort implementado armazena no arquivo 'filename' a escolha de pivôs realizdas durante as chamadas recursivas. A informação armazenada no aruivo não é o pivô propriamente dito, mas sim, sua posição relativa ao tamanho do vetor (índice do pivô/tamanho da lista).

Implementação 1 QuickSort

```
def partition(L):
    pivot = L[0]
    lower = 0
    for current in range (1,len(L)):
        if L[current] < pivot:
            lower +=1
            temp = L[lower]
            L[lower] = L[current]
            L[current] = temp
    L[0] = L[lower]
    L[lower] = pivot
    #Este for foi incluído para adaptar o algoritmo para entradas com números duplicados,
    #a complexidade aumenta de k*n para c*2n mas ainda é O(n)
    for current in range(lower+1,len(L)):
        if L[current] == pivot:
            lower +=1
            emp = L[lower]
            L[lower] = L[current]
            L[current] = temp
    return L, lower

def quicksort(L, filename):
    if len(L) < 2:
        return L
    else:
        (lst, pivot_index) = partition(L)
        fl=open(filename, 'a')
        fl.write(str(float(pivot_index+1)/float(len(L)))+'\n')
        return quicksort(lst[:pivot_index],filename)+[L[pivot_index]] +quick-
sort(lst[pivot_index+1:],filename)
```

2 Análise da Escolha dos Pivôs

2.1 Gerando arquivo de números aleatórios

Inicialmente iremos gerar um arquivo com com N*M números aleatórios, onde N será o número de rodadas do quicksort e M o número de elementos da lista que deverá ser ordenada em cada rodada. No método abaixo N= len_file e M= len_list:

Implementação 2 Métoto para Gerar Arquivo Temporário com Números Aleatórios

```
import random, tempfile

def modfile(len_file, len_list):
    with tempfile.NamedTemporaryFile(mode = 'w+', delete = False) as newFile:
        for i in range(0,len_file*len_list):
            newFile.write(str(random.randint(0, 1000))+'\n')
    return newFile.name
```

Para gerar o arquivo fazemos:

```
len_file = 3000 #Número N de rodadas
len_list = 1000 #Número M de números aleatórios
tempfile = modfile(len_file, len_list)
```

2.2 Recuperando listas do arquivo temporário e realizar chamadas do Quicksort

Implementação 3

```
f = open(tempfile)
lines = f.readlines()

#Recuperando listas do arquivo
lists = []
line = 0
list = 0
while line < len(lines):
    L=[]
    for i in range(0,len_list):
        L.insert(i,int(lines[line]))
        i+=1
        line += 1
    lists.insert(list, L)
    list+=1

#N rodadas sucessivas do quicksort
filename = 'pivots.txt'
for lst in lists:
    quicksort(lst, filename)
```

2.3 Recuperando informações sobre pivôs

Implementação 4

```
f = open(filename)
pivots_lines = f.readlines()
pivots = []
line = 0
while line < len(pivots_lines):
    pivots.insert(line,float(pivots_lines[line]))
    line += 1
```

2.4 Gerando tabela de análise dos pivôs

O script a seguir foi implementado para gerar um Pandas Dataframe com a informação sobre o pivô que corresponde à sua posição relativa na lista (citada na seção 1) e suas respectivas análises, indicando se foi uma escolha boa ou uma escolha ruim.

Implementação 5

```
import pandas as pd
df= pd.DataFrame(pivots)
df.columns = ['pivos']

def analysis(pivot):
    if pivot < 0.75 and pivot > 0.25:
        return 'boa'
    else:
        return 'ruim'

df['escolha'] = df['pivos'].apply(analysis)
df
```

A figura a seguir é uma amostra do dataframe obtido:

29	0.333333	boa
...	...	...
8267961	0.200000	ruim
8267962	1.000000	ruim
8267963	0.500000	boa
8267964	1.000000	ruim
8267965	0.545455	boa
8267966	0.083333	ruim
8267967	1.000000	ruim
8267968	0.333333	boa
8267969	0.812500	ruim
8267970	1.000000	ruim
8267971	0.500000	boa
8267972	0.200000	ruim
8267973	1.000000	ruim
8267974	0.333333	boa
8267975	0.666667	boa
8267976	0.909091	ruim
8267977	0.153846	ruim
8267978	1.000000	ruim
8267979	0.500000	boa
8267980	0.666667	boa
8267981	0.625000	boa
8267982	1.000000	ruim
8267983	1.000000	ruim
8267984	1.000000	ruim
8267985	0.200000	ruim
8267986	0.642857	boa
8267987	0.500000	boa
8267988	0.377778	boa
8267989	0.196429	ruim
8267990	0.944000	ruim

8267991 rows × 2 columns

Figura 1: Tabela de análise dos pivôs

Finalmente, calculamos a porcentagem de boas escolhas de pivôs como segue:

Implementação 6 Analisando porcentagem de boas escolhas

```
count = 0
for ratio in df['pivos']:
    if ratio < 0.75 and ratio > 0.25:
    count+=1
print float(count)/float(len(pivots))
```

O resultado obtido foi de aproximadamente 48% de boas escolhas para 3000 rodadas em listas

de 1000 números aleatórios. Esta porcentagem é suficientemente razoável para garantir O(n log n) no Quicksort.