Projeto COVID 19
Algoritmos e Tipo Abstratos de Dados
Instituto Politécnico de Setúbal
Bruno Silva
Patrícia Macedo
LEI_01

Hysa Mello de Alcântara *190221039*

Rafael da Rosa Marçalo *190221048*

30/6/2020

Índice

Descrição dos ADT's

ADTList

O *ADTList* foi utilizado para guardar as informações de pacientes de forma ordenada. A implementação escolhida foi a de *ArrayList* por ser mais prática e eficiente, tendo complexidade algorítmica mais baixa para as funções utilizadas, de forma a que o código fosse o mais rápido e simples possível. Desta forma, fazendo os comandos a serem corridos mais facilmento. Em comparação, com outro tipo de ADT, se fosse utilizada a implementação em *LinkedList*, a maioria das complexidades seriam **O(n)**, ao contrário do ArrayList que tem algumas funções em **O(1)**.

ADTMap

O *ADTMap* foi utilizado para guardar as regiões, tendo em conta o nome da mesma como *MapKey* e a região em si como *MapValue*. Assim como no *ADTList*, a implementação utilizada foi a de *ArrayList* por motivos de simplicidade de código. Além disso, ambas as implementações possíveis apresentavam em geral a complexidade **O(n)** nas suas funções, logo escolheu se a mais simples.

Complexidades

Comandos Base

- LOADP - Complexidade O(n);
- LOADR - Complexidade O(n);

Indicadores Simples e Pesquisas

- AVERAGE - Complexidade O(n);
- FOLLOW - Complexidade $O(n^2)$;
- SEX - Complexidade O(n);
- SHOW - Complexidade O(n);
- TOP5 - Complexidade O(n);
- OLDEST - Complexidade O(n);
- GROWTH - Complexidade O(n);
- MATRIX - Complexidade O(n);

Indicadores Avançados

- REGIONS - Complexidade $O(n^2)$;
- REPORT - Complexidade $O(n^3)$;

Pseudo-Código

Sex

```
Algorithm sex
    input: *list - PtList

BEGIN

size <- 0, fem <- 0,  male <- 0, ukn <- 0
listSize(\*list, &size)
listElem pat
FOR i <- 0 to size-1 DO
    listGet(\*list, i, &pat)
    IF strcmp(pat.sex, "female") = 0 THEN
        fem <- fem + 1
    ELSE IF strcmp(pat.sex, "male") = 0 THEN
        male <- male + 1
    ELSE
        ukn++
    END IF
END FOR

PRINT "Percentage of Females: "$(round(fem * 100/size))""
PRINT "Percentage of Males: "$(round(male * 100/size))""
PRINT "Percentage of Unknown: "$(round(ukn * 100/size))""
PRINT "Total: $size"

END
```

Matrix

```
Algorithm matrixGenerator
    input: *list - PtList

BEGIN

IF list != NULL THEN
    *fieldsOne <- ageGapArray(list, 0, 15)
    *fieldsTwo <- ageGapArray(list, 16, 30)
    *fieldsThree <- ageGapArray(list, 31, 45)
    *fieldsFour <- ageGapArray(list, 46, 60)
    *fieldsFive <- ageGapArray(list, 61, 75)
    *fieldsSix <- ageGapArray(list, 76, $INVALID\_FIELD)

    PRINT "|Age Gap\t|Isolated\t|Deceased\t|\t|\n"
    PRINT "|[0-15]\t|$fieldsOne[0]\t|$fieldsOne[1]\t|$fieldsOne[2]\t|\n"
    PRINT "|[16-30]\t|$fieldsTwo[0]\t|$fieldsTwo[1]\t|$fieldsTwo[2]\t|\n"
    PRINT "|[31-45]\t|$fieldsThree[0]\t|$fieldsThree[1]\t|$fieldsThree[2]\t|\n"
    PRINT "|[46-60]\t|$fieldsFour[0]\t|$fieldsFour[1]\t|$fieldsFour[2]\t|\n"
    PRINT "|[61-75]\t|$fieldsFive[0]\t|$fieldsFive[1]\t|$fieldsFive[2]\t|\n"
    PRINT "|[76-...]\t|$fieldsSix[0]\t|$fieldsSix[1]\t|$fieldsSix[2]\t|\n"

    free(fieldsOne)
    free(fieldsTwo)
    free(fieldsThree)
    free(fieldsFour)
    free(fieldsFive)
    free(fieldsSix)
END IF

END
```

Regions

```
Algorithm regionsAlphaOrder
    input: *regions - PtMap
           *patients - PtList


BEGIN

pSize <- 0
listSize(\*patients, &pSize)
ListElem el1

mSize <- 0
mapSize(\*regions, &mSize)

\*mkaux <- (MapKey \*) calloc(mSize + 1, sizeOf(MapKey))
\*mk <- mapKeys(\*regions)

FOR i <- 0 TO mSize DO
    for j <- 0 TO pSize -1 DO
        listGet(\*patients, j, &el1)
        IF strcmp(el1.region, mk[i].str) = 0 THEN
            IF strcmp(el1.status, "isolated") = 0 THEN
                mkaux[i] <- mk[i]
            END IF
        END IF
    END FOR
END FOR

qsort(mkaux, mSize, sizeof(MapKey), compareTo)

FOR i <- 0 TO mSize DO
    keyStringPrint(mkaux[i])
    PRINT "\n"
END FOR

free(mk)
free(mkaux)

END
```

Conclusão

Este projeto foi bastante interessante pois permitiu que os nossos conhecimentos relativos à linguagem C fossem aprofundados de forma mais concreta, tendo em vista a complexidade mesma. Desta forma, o projeto ajudou-nos a perceber e interpretar como iríamos manipular os dados desta específica maneira através de funções encadeadas e coesas e até mesmo pseudo-código que se desenvolveu para algumas funções.

Com a ajuda da docente da disciplina nas sessões síncronas e por e-mails, conseguimos ser bem orientados na organização do mesmo, sendo a sua ajuda importante ao desenvolvimento do projeto. Desta forma, concluímos os nossos objetivos de manipular ADTs e ficheiros com sucesso, fazendo o possível para eliminar quaisquer tipos de erros e altas complexidades algorítmicas que não permitissem o código ser bem sucedido. Portanto, em geral podemos constatar que o projeto foi importante para a consolidação de conhecimentos da linguagem C, e que, desta forma, estamos preparados para utilizá-lo de forma eficiente em outras situações de necessidade.