23 Aula 22: 29/OUT/2019

23.1 Aulas anteriores

Na aula 20 estudamos aplicações de ordenação, busca binária e dicionário para desenvolver algoritmos para o problema `3SUM`:

- força-bruta que consume tempo $O(n^3)$
- ordenação com busca binária que consumia tempo $O(n^2 \lg n)$
- usando dicionários e consumia tempo **esperado** $O(n^2)$

Há ainda um algoritmo que usava ordenação e consume tempo $O(n^2)$ mas ãno discutimos.

Já na última aula vimos como resolver o problema da subsequência comum mais longa (*Longest Common Substring* = LCS) utilizando **programação dinâmica** (= recursão com tabela).

23.2 Hoje

Resolveremos alguns problemas que utilizam o tipo nativo `set` do Python. Também entrarão em ação as já conhecidas classes nativas `list` (lista), `dict` (discionário) e a já usada ordenação nativa `sort()` da classe `list`.

O ponto é que `set`, como `dict`, consome **tempo esperado** constante para as operações básicas como `x in set`.

23.3 Arquivos

Os problemas que veremos aparecem no *IntroCS* de Robert Sedgewick e Kevin Wayne da disciplina COS126 de Princeton.

Os programas no diretório `py` são:

- `dedup_list.py`: lê uma lista de palavras e imprime as palavras que ocorrem na lista, sem duplicatas. A solução utiliza `list` do Python.
- `dedup_setpy`: lê uma lista de palavras e imprime as palavras que ocorrem na lista, sem duplicatas. A solução utiliza `set` do Python e é muito mais eficiente que anterior.
- `spellchecker.py`: lê uma lista de palavras de um dicionário e depois lê um texto identificando todas as palavras do texto que não estão no dicionário. A solução utiliza `set` do Python para armazenar as palavras do dicionário.
- `anagramas.py`: lê uma lista de palavras e imprime grupos de palavras que são anagramas umas das outras. A solução utiliza um dicionário em que as chaves são strings e os valores são `set` de strings.

Para brincarmos com os programas temos alguns arquivos no diretório `txt`. A maioria foi copiada do *IntroCS* ou *algs4* de Robert Sedgewick e Kevin Wayne usados nas diciplinas COS126 e COS226 de Princeton.

- `pwords.txt`: lista de palavras de um dicionário português (não sabemos de onde veio)
- `words.utf-8.txt`: lista de palavras de um dicionário inglês
- `misspellings.txt`: palavras em inglês comumente escritas com erros tipográficos. Não estamos usando. Seria útil para fazermos um corretor ortográfico com sugestão de palavras.
- `dickens.txt`: texto de Charles Dickens, é um arquivo maior
- `bible.txt`: bíblia

- `lusiadas.txt`: pequeno trecho copiado de *Os Lusiadas* de Luís de Camões
- `amendments.txt`: emendas da constituição dos Estados Unidos da América

23.4 Bibliotecas

As soluções usam algumas bibliotecas

```
# para cronometrar o tempo gasto
import time

# constantes
import string
# >>> string.punctuation
# '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'

# usa re.sub("["+ string.punctuation+"]", ' ', texto) para substituir pontuação por " "
import re
```

23.5 Duplicadas

Problema. Escreva um programa que leia uma lista de palavras e imprima as palavras que ocorrem na lista sem repetir palavras.

23.5.1 `main()`

```
#--------------------------------------------------------------------------
def main():
    # leia o texto
    nome_arq = input("Digite o nome do arquivo com o texto: ")
    with open(nome_arq, "r", encoding="utf-8") as arq:
        texto = arq.read()

    # substitui pontuação por " "
    texto = re.sub("["+string.punctuation+"]", " ", texto)

    # crie lista de palavras
    lista_pals = texto.split()

    # crie lista com palavras sem repetições
    pals = sem_repetidos(lista_pals)

    print("main(): mostra dom 20 palavras no texto:")
    i = 0
    for pal in pals:
        print(pal)
        i += 1
        if i == 20: break
```

23.5.2 Solução com `list`

```
#-----------------------------------------------------------------
def sem_repetidos(lst):
    '''(lst) -> lst
    Retorna uma lista com o itens em lst sem repetições
    '''
    lst_pals = []
    for pal in lst:
        if pal not in lst_pals: # O(n) no pior caso, onde n é o número de palavras em lst_pa
            lst_pals.append(pal) # O(1)
    return lst_pals
```

Consumo de tempo é $\texttt{O}(\texttt{n}^2)$ onde `n` é o número de palavras lidas.

A solução mantendo a lista ordenada continuaria a consumir tempo quadrático.

Experimento

```
python dedup_list.py < ../txt/bible.txt
main(): lendo o texto ...
main(): texto lido ...
main(): removendo pontuação ...
main(): pontuação removida ...
main(): criando lista de palavras ...
main(): lista com 767855 palavras criada ...
main(): criando lista de palavras ...
main(): lista com 13456 palavras criado
elapsed time     = 10.643
main(): mostra dom 20 palavras no texto:
In
the
beginning
God
created
heaven
and
earth
And
was
without
form
void
darkness
upon
face
of
deep
Spirit
moved
```

23.5.3 Solução com `dict`

A parte relevante da solução é

```
#-------------------------------------------------------------------
def sem_repetidos(lst):
    '''(lst) -> set
    Retorna um conjunto com o itens em lst sem repetições
    '''
    # pals = {pal for pal in lst} # alternativamente
    # pals = list(lst) # alternativamente
    set_pals = set()
    for pal in lst:
        set_pals.add(pal) # O(1) esperado
    return set_pals
```

Consumo de tempo **esperado** é `O(n)` onde `n` é o número de palavras lidas.

23.5.4 Experimento com `set`

```
% python dedup_set.py < ../txt/bible.txt
main(): lendo o texto ...
main(): texto lido ...
main(): removendo pontuação ...
main(): pontuação removida ...
main(): criando lista de palavras ...
main(): lista com 767855 palavras criada ...
main(): criando conjunto de palavras ...
main(): conjunto com 13456 palavras criado
elapsed time     = 0.057
main(): mostra dom 20 palavras no texto:
dew
genealogy
walkedst
damsel
amethyst
deck
ghost
Philologus
similitudes
drink
Persecutions
press
Shimshai
continueth
betrayers
Reubenites
pacified
worldly
swellings
locks
```

23.6 Verificador ortográfico

Problema. Escreva um programa que leia uma lista de palavras de um dicionário e em seguida leia um texto. O programa deve indicar todas as palavras do texto que não aparecem na lista de palavras do dicionário.

Solução

```
#---------------------------------------------------------------------
def main():
    # leia as palavras corretas
    set_pals = set()
    with open("../txt/words.utf-8.txt", "r", encoding="utf-8") as arq:
        for line in arq:
            pal = line.strip().lower()
            set_pals.add(pal)

    # leia texto
    nome_arq = input("Digite o nome do arquivo com o texto: ")
    with open(nome_arq, "r", encoding="utf-8") as arq:
        texto = arq.read()

    # verifique a ortografia das palavras no texto
    linhas_texto = texto.split("\n")
    no_lin = 0
    conta  = 0
    for linha in linhas_texto:
        linha = re.sub("["+string.punctuation+"]", " ", linha) # substitui pontuação por " "
        lista_pals = linha.split()
        for pal in lista_pals:
            pal = pal.lower()
            if pal != "" and pal not in set_pals:
                print("%s: %s"%(no_lin, pal))
                conta += 1
        no_lin += 1
    fim = time.time()
    print("main(): foram encontradas %s palavras desconhecidas"%conta)
```

23.6.1 Experimento

```
% python spellchecker.py < ../txt/dickens.txt
main(): lendo palavras no dicionário ...
main(): dicionário criado com 616524 palavras
main(): lendo texto e verificando a ortografia...
21: southcott
72: turnham
210: offensive
299: tellson
300: tellson
433: tellson
445: tellson
453: tellson
455: tellson
477: inquired
572: disageeable
...
592400: pirrip
592460: gardenwalk
main(): verificação encerrada.
main(): foram encontradas 48317 palavras desconhecidas
elapsed time     = 3.765
```

23.7 Anagramas

Problema. Escreva um programa que leia uma lista de palavras e imprima a lista com as palavras que são anagramas.

Exemplo:

```
% more pt.in
vixe
louro
tiro
orlou
rolou
rito
trio

% python anagramas.py < pt.in
* rito tiro trio
* louro orlou rolou
```

23.7.1 Ideia

Usar um dicionário em que anagramas tenham a mesma chave. O conjunto de anagramas pode ser o valor de uma tal chave.

Como chave podemos utilizar as letras que apararecem na palavra **ordenadas alfabeticamente**. Assim `'eivx'` será a chave de `'vixe'`, `'looru'` será a chave de `'louro'`, `'orlou'` e `'roulou'`.

```
% python anagramas.py < ../txt/pt.in
main(): lendo as palavras ...
main(): palavras lidas ...
main(): criando lista de palavras ...
main(): lista com 7 palavras criada ...
main(): criando conjunto de palavras ...
main(): dicionário com 3 anagramas criado
elapsed time     = 0.000
main(): mostra os anagramas:
looru: orlou rolou louro
iort: trio rito tiro
```

23.7.2 `main()`

Utiliza um dicionário em que as chaves são strings e os valores são `set` de strings.

```
#-------------------------------------------------------------------------
def main():
    # leia o texto
    nome_arq = input("Digite o nome do arquivo com as palavras: ")
    with open(nome_arq, "r", encoding="utf-8") as arq:
        pals = arq.read()
```

```
    lst_pals = pals.split()


    # crie dicionário de de anagramas
    dicio_anagramas = anagramas(lst_pals)
    print("main(): dicionário com %s anagramas criado"%(len(dicio_anagramas)))

    print("main(): mostra os anagramas:")
    for chave in dicio_anagramas:
        if len(dicio_anagramas[chave]) > 1:
            print("%s: "%chave, end="")
            for pal in dicio_anagramas[chave]:
                print(pal, end=' ')
            print() # muda de linha
```

Dicionário de anagramas

```
def anagramas(lst_pals):
    ''' (list) -> dict
    Retorna um dicionário em que cada chave é uma string
    e o correspondente valor é o conjunto de anagramas da
    string.
    '''
    dicio_anagramas = {}
    for pal in lst_pals:
        # letras na palavra
        pal_aux = list(pal)

        # letras ordenadas
        pal_aux.sort()

        # cria a chave do dicionário
        chave = ''.join(pal_aux)

        # se a chave não está no dicionário insira
        if chave not in dicio_anagramas: # O(1) esperado
            dicio_anagramas[chave] = set()

        # coloque o anagrama no cojunto da chave
        dicio_anagramas[chave].add(pal)

    return dicio_anagramas
```

Experimentos

```
% python anagramas.py < ../txt/pt.in
main(): lendo as palavras ...
main(): palavras lidas ...
main(): criando lista de palavras ...
main(): lista com 7 palavras criada ...
main(): criando conjunto de palavras ...
main(): dicionário com 3 anagramas criado
elapsed time     = 0.000
main(): mostra os anagramas:
looru: orlou rolou louro
iort: trio rito tiro


% python anagramas.py < ../txt/words5.txt
main(): lendo as palavras ...
main(): palavras lidas ...
main(): criando lista de palavras ...
main(): lista com 2415 palavras criada ...
main(): criando conjunto de palavras ...
main(): dicionário com 2188 anagramas criado
elapsed time     = 0.014
main(): mostra os anagramas:
abdeo: adobe abode
adnor: radon adorn
aeegr: eager agree
aelrt: alter later alert
alloy: alloy loyal
aflot: aloft float
abemr: amber bream
abelm: amble blame
adeim: amide media
aelmp: ample maple
aegln: glean angel angle
aegnr: anger range
agnry: rangy angry
agnor: organ groan argon
agort: groat argot gator
aegru: auger argue
aeirs: arise raise
anors: arson sonar
aerst: stare aster
acitt: tacit attic
aalru: aural laura
aciru: curia auric
abegl: bagel gable
aabls: basal balsa
aabln: banal nabla
```

abiss: basis bassi
abest: baste beast
abdey: beady bayed
abder: bread beard debar
begin: binge begin being
below: bowel elbow below
abelt: table bleat
abdor: broad board
abdir: rabid braid
abekr: brake **break**
bhort: throb broth
bnrtu: burnt brunt
bhrsu: brush shrub
bertu: brute rebut buret
bdegu: budge debug
beglu: bugle bulge
acder: cadre cedar
aceno: ocean canoe
acnst: scant canst
acert: caret carte crate cater trace
acenr: carne crane
abcor: carob cobra
aclor: carol coral
acerv: crave carve
acesu: sauce cause
achin: china chain
achmr: charm march
acehp: cheap peach
aceht: teach cheat
cehrt: chert retch
cefhi: fiche chief
cehin: chine niche
cdeir: cider cried
aciil: iliac cilia
ceitv: evict civet
aclps: clasp scalp
aceln: clean lance
acelt: eclat cleat
cdlou: could cloud
cnory: corny crony
cdeor: credo decor
ceepr: creep crepe
celru: cruel lucre ulcer
censt: scent csnet
adiry: diary dairy
adert: trade tread dater
adelt: delta dealt
deefr: freed defer
defiy: edify deify
deity: diety deity

dginy: dying dingy
degir: dirge ridge
dnoor: rondo donor
dorwy: wordy dowry rowdy
adinr: nadir drain
adepr: padre drape
dstuy: study dusty
aehrt: heart hater earth
aeels: easel lease
eegrt: egret greet
eipqu: equip pique
eerst: steer terse ester
eehrt: there ether three
ehost: those ethos
eervy: veery every
aeltx: latex exalt
afilr: flair frail
efhls: flesh shelf
efilr: flier rifle
fhort: froth forth
eefrr: refer freer
aagmm: magma gamma
eegnr: genre green
ghirt: right girth
aglns: slang glans
aeglr: glare large lager regal
aegrt: great grate
gnorw: wrong grown
gstuy: gutsy gusty
adhry: hydra hardy
ehors: shore horse
ainpt: paint inapt
cinru: runic incur
einrt: inter inert
einst: inset stein
einru: urine inure
adekn: knead naked
aelps: sepal lapse
alstu: latus talus
aelpt: petal pleat plate leapt
aelnr: learn renal
aehls: leash shale
aelst: steal slate stale least
elmno: monel lemon melon
eelpr: repel leper
eelrv: lever revel
aemrs: smear maser
eimrt: mitre merit remit
almor: moral molar
admno: nomad monad

emott: totem motet
ansty: tansy nasty
aelnv: venal navel
eenrv: nerve never
ghint: thing night
hnort: north thorn
enost: onset stone
ciopt: topic optic
ghotu: ought tough
alpsy: splay palsy
aelnp: plane panel penal
aeprs: spear parse spare
aepst: paste septa spate
aeprt: pater taper
adelp: pedal plead
einps: snipe penis spine
eepst: peste steep
eiprt: petri tripe
aehps: shape phase
inopt: point pinto
almps: plasm psalm
eopsy: posey poesy
ilops: polis spoil
eoprs: spore prose
eprsu: super sprue purse
eiqtu: quite quiet
aelrv: ravel velar
einrs: resin siren risen rinse
egoru: rouge rogue
oorst: torso roost
ainst: saint stain satin
aelsv: slave salve
arsty: satyr stray
cinos: scion sonic
enors: snore senor
eersv: verse serve sever
epstu: upset setup
aehrs: shear share
eehst: sheet these
hoost: shoot sooth
hostu: shout south
deils: slide sidle
ilsty: silty styli
einsw: sinew swine
aekst: steak stake skate
ailns: snail slain
eelst: sleet steel
eilms: smile slime
loops: sloop spool
aekns: snake sneak

noops: snoop spoon
ilpst: spilt split
oprst: sport strop
aestt: taste state
arstw: swart straw
rsttu: trust strut
aestw: sweat waste
eorsw: swore worse
alnot: talon tonal
aertt: treat tater
hortw: throw worth
adoty: today toady
eortw: tower wrote
ailrt: trial trail
orttu: trout tutor
elovw: vowel wolve