



UNIVERSIDADE ESTADUAL DE SANTA CRUZ – UESC

ÍCARO MARADEI COSTA BORGES

**SISTEMA PARA ANÁLISE DE PROTEÍNAS: OTIMIZAÇÃO NO
PROCESSAMENTO DE ARQUIVOS E REGIÕES INTRINSECAMENTE
DESORDENADAS**

**ILHÉUS - BAHIA
2019**

ÍCARO MARADEI COSTA BORGES

**SISTEMA PARA ANÁLISE DE PROTEÍNAS: OTIMIZAÇÃO NO
PROCESSAMENTO DE ARQUIVOS E REGIÕES INTRINSECAMENTE
DESORDENADAS**

Trabalho de Conclusão de Curso
apresentado à Universidade Estadual de
Santa Cruz - UESC, como parte das
exigências para obtenção do título de
Bacharel em Ciência da Computação.

Orientador: Prof. Paulo Eduardo Ambrósio

ILHÉUS - BAHIA
2019

ÍCARO MARADEI COSTA BORGES

**SISTEMA PARA ANÁLISE DE PROTEÍNAS: OTIMIZAÇÃO NO
PROCESSAMENTO DE ARQUIVOS E REGIÕES INTRINSECAMENTE
DESORDENADAS**

Trabalho de Conclusão de Curso
apresentado à Universidade Estadual de
Santa Cruz - UESC, como parte das
exigências para obtenção do título de
Bacharel em Ciência da Computação.

Ilhéus, 19 de Agosto de 2019.

Banca Examinadora

Prof. Paulo Eduardo Ambrósio
UESC/DCET
(Orientador)

Prof. _____
UESC/DCET

Prof. _____
UESC/DCET

SISTEMA PARA ANÁLISE DE PROTEÍNAS: OTIMIZAÇÃO NO PROCESSAMENTO DE ARQUIVOS E REGIÕES INTRINSECAMENTE DESORDENADAS

RESUMO

As pesquisas no ramo da bioinformática estão se tornando cada vez mais acessíveis e práticas. No entanto, ainda há alguns *softwares* que necessitam de uma otimização melhor dos dados, pois estes utilizam-se de bancos disponíveis *online* para fazer verificação, comparação e processamento. Tendo isso em vista, este trabalho visa prover ferramentas para o auxílio na diminuição da exigência de poder das máquinas, com base em análise e/ou manipulação de arquivos que representem as proteínas de forma computacional. Foi desenvolvido um sistema *web* para o tratamento das informações biotecnológicas específicas recebidas, com uma reorganização automatizada dos dados com base em parâmetros passados através de ferramentas como a linguagem de programação *Python* e o *microframework web Flask*. A manipulação dos dados mostrou-se bastante precisa e bem desempenhada, com a distribuição da informação sendo balanceada e com o sistema apresentando um bom tratamento de erros e nenhuma falha até então. Além disso, foi desenvolvido um script, com base em resultados do programa voltado para a biotecnologia DISOPRED, voltado para as chamadas Regiões Intrinsecamente Desordenadas, que tratam-se de trechos das proteínas que possuem estudos relativamente recentes e há grande busca por conhecimento científico nesta área.

Palavras-chave: Proteínas, bioinformática, otimização

LISTA DE FIGURAS

Figura 1	16
Figura 2	17
Figura 3	20
Figura 4	21
Figura 5	21
Figura 6	22
Figura 7	24
Figura 8	25
Figura 9	27
Figura 10	28

LISTA DE TABELAS

Tabela 1	14
Tabela 2	26
Tabela 3	27

LISTA DE SIGLAS

CACAU	Centro de Armazenamento de dados e Computação Avançada da UESC
CSS	<i>Cascading Style Sheets</i>
HTML	<i>HyperText Markup Language</i>
IDP	<i>Intrinsically Disordered Proteins</i>
IDR	<i>Intrinsically Disordered Regions</i>
NCBI	<i>National Center for Biotechnology Information</i>
UESC	Universidade Estadual de Santa Cruz
URL	<i>Uniform Resource Locator</i>

SUMÁRIO

1	INTRODUÇÃO	11
1.1	Objetivos	12
1.1.1	Gerais.....	12
1.1.2	Específicos	12
2	REFERENCIAL TEÓRICO.....	13
2.1	Proteínas	13
2.2	Proteínas Intrinsecamente Desestruturadas.....	15
2.3	Bioinformática	16
3	MATERIAIS E MÉTODOS	19
3.1	Python/<i>Sublime Text 2</i>	19
3.2	<i>Flask</i>	20
3.3	HTML, CSS & Bootstrap.....	22
3.4	Outros <i>softwares</i>	23
3.5	Funcionamento do projeto	23
3.5.1	Desenvolvimento local.....	23
3.5.2	Desenvolvimento do <i>script</i> dentro do <i>website</i>	24
3.5.3	Desenvolvimento <i>Web</i>	25
4	RESULTADOS E DISCUSSÕES	26
5	CONSIDERAÇÕES FINAIS	29
6	TRABALHOS FUTUROS.....	30
	REFERÊNCIAS.....	31

1 INTRODUÇÃO

A bioinformática é um dos ramos com mais conteúdo na área da tecnologia da informação, visto que ambas podem ser aliadas no que diz respeito à pesquisa e no auxílio de criação de ferramentas, seja voltada para a biologia, como por exemplo, uma análise *in silico* (de forma computacional) de uma espécie, ou dedicada à computação, tal como são desenvolvidos os algoritmos evolutivos. E, mesmo dentro do próprio âmbito biológico, há uma infinidade de subáreas onde atividades podem ser desenvolvidas.

A nível molecular, os ramos da biotecnologia mais conhecidos popularmente estão ligados ao DNA e todo o estudo no que diz respeito ao seu funcionamento e seus mecanismos. No entanto, há várias outras áreas a serem trabalhadas, sendo uma delas o conhecimento relacionado às proteínas. Há um conhecimento geral com relação a estas macromoléculas que representa pouco em comparação com a sua grande variedade de aplicação.

As proteínas são macromoléculas formadas por um conjunto de aminoácidos ligados entre si, sendo uma substância presente em todos os processos biológicos e, portanto, imprescindível para a vida de qualquer espécie. Também chamadas de cadeias polipeptídicas, elas possuem uma vasta quantidade de funções no organismo, e, para isso, precisam ter uma gama extensa de formatos, combinações e composições. E é o que acontece.

Quando se trata de análise computacional, as proteínas podem ser representadas em modelos 3D, através de *softwares* de renderização, e, sendo desta forma, é possível visualizar até suas estruturas quaternárias (onde as proteínas se ligam e formam um complexo multi-proteico maior). No entanto, para alguns tipos de análise, não é necessária a visualização em três dimensões; basta obter informações da sua estrutura primária (combinação de aminoácidos) para se trabalhar com os dados obtidos, e é nisso que este trabalho foca.

A cadeia de aminoácidos é representada de uma forma específica computacionalmente, chamada arquivo *fasta*. Com base nesse tipo de dado, é possível trabalhar com as informações já obtidas em alguns bancos de dados *online*, existindo alguns *softwares* que trabalham com esse tipo de arquivo, fazendo, por exemplo, a predição das funções de cada trecho da proteína. Contudo, o processamento costuma ser demorado, pois é feita uma comparação com dados

disponíveis publicamente e esses são conferidos com os que estão em análise, para assim emitir um resultado. Com base nisso, é de grande ajuda que, tendo na Universidade Estadual de Santa Cruz um supercomputador capaz de executar vários processos simultaneamente, exista uma ferramenta capaz de balancear o máximo possível para cada núcleo disponível.

Além disso, há um pequeno foco deste trabalho no que diz respeito à análise do programa DISOPRED, que prediz onde estão as IDRs das proteínas, porém emitindo como saída um arquivo com difícil observação. De forma simples, é possível extrair esta informação para tornar-se mais legível.

1.1 Objetivos

1.1.1 Gerais

Desenvolver mecanismos para melhorar a análise de dados computacionais das proteínas, facilitando e balanceando o seu desempenho no supercomputador CACAU da Universidade Estadual de Santa Cruz, ou aprimorando a visualização de dados de saída de outros programas da área da bioinformática.

1.1.2 Específicos

- a) Desenvolver um sistema *web* capaz de processar um arquivo de representação de proteína
- b) Encaminhar o resultado deste processamento de forma mais direta para ser processado pelo CACAU
- c) Desenvolver um *script* para ser possível fazer uma análise mais concisa com relação às IDRs

2 REFERENCIAL TEÓRICO

2.1 Proteínas

As proteínas são as substâncias macromoleculares mais imprescindíveis à nossa vida. São substâncias que estão o tempo todo sendo produzidas, consumidas, utilizadas, entre outros. Estes polímeros podem trabalhar nas mais variadas funções e, entre algumas delas, segundo SANTOS (2019), no “[...] transporte de oxigênio (hemoglobina), na proteção do corpo contra organismos patogênicos (anticorpos), como catalizadora de reações químicas (enzimas), receptora de membrana, atuação na contração muscular (actina e miosina), além de serem fundamentais para o crescimento e formação dos hormônios”.

Essa grande variedade de funções deve-se à forma em que a proteína é construída. Quando diz respeito à sua composição inter-substancial, ou seja, a forma como suas composições se ligam, as proteínas podem ter 4 níveis: estrutura primária, secundária, terciária e quaternária.

Indo da maior à menor, a estrutura quaternária é a ligação de proteínas umas com as outras, formando polímeros cada vez maiores. A terciária diz respeito ao conjunto de peptídeos menores, segundo CELI (2019) “correspondendo ao dobramento (sobre ela mesmo) da cadeia polipeptídica” e “é caracterizada pelo formato tridimensional específico”. A secundária trata-se de pequenos trechos internos da proteína, onde ela pode adotar mais comumente apenas dois tipos de formato (alfa-hélice e folha-beta).

Já a estrutura primária será o alvo deste trabalho. Este nível trata-se da sequência de aminoácidos ligados entre si, pois é isso que a proteína é, em resumo: uma sequência de aminoácidos. Cada proteína é formada por, segundo Pinheiro (2019), no mínimo, 50 aminoácidos interligados, podendo ter proteínas no corpo humano que passam dos 30 mil aminoácidos em sua composição. E uma das suas formas de representação *in silico*, ou seja, através de um ambiente computacional, é apenas uma sequência de letras.

Computacionalmente, cada aminoácido é representado por uma letra do alfabeto (existem 20 aminoácidos conhecidos e catalogados até então, excluindo as letras B, “J”, “O”, “U”, “X” e “Z” da representação), e uma proteína, sendo uma sequência de aminoácidos, pode ser representada como uma sequência de letras,

separadas, em cada ligação de uma proteína menor com outra, pelos chamados cabeçalhos.

Tabela 1 – Representação dos aminoácidos

Letra representante	Nome do aminoácido
A	Alanina
C	Cisteína
D	Ácido Aspártico
E	Ácido Glutâmico
F	Fenilalanina
G	Glicina
H	Histidina
I	Isoleucina
K	Lisina
L	Leucina
M	Metionina
N	Asparagina
P	Prolina
Q	Glutamina
R	Arginina
S	Serina
T	Treonina
V	Valina
W	Triptofano
Y	Tirosina

Fonte: Elaborada pelo autor

A análise *in silico* de proteínas permite que sejam descobertas as suas funções sem a necessidade de ter acesso à molécula em si, com base apenas na sua representação e na sua sequência de aminoácidos. Por mais que haja uma infinidade de possibilidades de arranjo para formar novas proteínas, há uma certa previsibilidade em seus padrões, podendo ser definida a função que a proteína desempenha de

acordo com sua sequência de aminoácidos em certa região. Além disso, com relação aos trechos das proteínas, há outro tipo de análise que pode ser feita.

2.2 Proteínas Intrinsecamente Desestruturadas

Até alguns anos atrás, as proteínas eram definidas por serem estáticas e possuírem o formato adequado para a interação com outra substância em específico, modelo chamado de chave-fechadura. No entanto, com o passar dos anos, foi descoberto que a maioria das cadeias de aminoácidos possuem as chamadas Regiões Intrinsecamente Desordenadas (IDR, do inglês *Intrinsically Disordered Regions*), que se ligam com outros substratos de forma dinâmica, podem se moldar de acordo com a situação ou até desempenhar mais de uma função.

Esta descoberta foi feita segundo Li (2015) em 1978, através de cristalografia de raios X e espectroscopia por ressonância magnética nuclear, quando determinadas proteínas não se comportaram de acordo com o esperado e mostraram que podem mudar sua forma em algumas regiões, trechos estes chamados de Intrinsecamente Desordenados. As proteínas que possuem esta característica em quantidade majoritária são chamadas de IDP, do inglês *Intrinsically Disordered Proteins*

Segundo Meng (2017, tradução nossa), “de acordo com estimativas recentes, trechos desordenados de aminoácidos compreendem 19% do total destas substâncias em proteínas eucarióticas, 6% em proteínas bacterianas”, além disso, “em eucariontes, 30% a 50% das proteínas possuem ao menos uma IDR longa (onde mais de 30 aminoácidos em sequência são intrinsecamente desordenados) e 6% a 17% das proteínas são preditas a serem completamente desordenadas. Por fim, 44% dos genes humanos codificam proteínas com IDRs longas” (Oates et al., 2013 *apud* Meng et al., 2017)

Portanto, uma importante área de estudo é a deste tipo de proteínas, visto que ainda não se sabe completamente como funcionam a mudança de função nesta região em proteínas. Para isso, alguns ramos da bioinformática auxilia com o desenvolvimento e uso constante de *softwares* de predição de IDPs e de predição de funções de proteínas. Combinando os dois dados, é possível associar se há uma grande influência das IDRs nas funções primordiais das proteínas.

2.3 Bioinformática

A bioinformática ajuda diretamente nestes dois segmentos. No que diz respeito às proteínas em si, há toda uma padronização do estudo e trabalho em cima destas substâncias de forma computacional, e dezenas de bancos de dados disponíveis *online* com informações dos mais variados tipos.

As proteínas, como dito anteriormente, são representadas por uma sequência de letras separadas por um cabeçalho. Os arquivos possuem uma extensão em específico, e são chamados de arquivos *fasta*, onde os dados são representados, por exemplo, de acordo com a imagem a seguir:

Figura 1 – Início do arquivo *fasta* que representa o genoma humano codificado em proteínas

```
human.faa
1 >NP_000005.2 alpha-2-macroglobulin isoform a precursor [Homo sapiens]
2 MGKNKLLHPSLVLLLLLPTDASVSGKPQYMLVPSLLHTEETTEKGCVLLSYLNETVTVSASLESVRGNRSLFTDLEAE
3 NDVLHCVAFAPKSSSNEEVMFLTVQVKGPTQEFKKRTTVMKNEDSLVFVQTDKSIYKPGQTVKFRVVSMDENFHPLE
4 LIPLVYIQDPKGNRIAQWQSFQLEGGKQFSPFLSSEPFQGSYKVVVQKKSGGRTEHPFTVEEFVLPKFEVQVTVPKIIT
5 ILEEEVMVSVCGLYTYGKPPVGHVTVSICRKYSDASDCHGEDSQAFCEKFSGQLNSHGCFYQQVKTQVFQVKRKEYEMKL
6 HTEAQIQEEGTVVELTGRQSSSEITRTITKLSFVKVDSHFROGIPFFGQVRLVDGKGVPINPKVIFIRGNEANYYSNATTD
7 EHGLVQFSINTTNVMGTSLTVRVNYKDRSPCYQWVSEEEHAAHTAYLVFSPSKSFVHLEPMISHELPCGHTQTQAHY
8 ILNGGTLGLKKLSFYLLIMAKGGIVRTGTHGLLVKQEDMKGHFSISIPVKSDIAPVARLLIYAVLPTGDIVIGDSAKYDV
9 ENCLANKVDLSFSPSQSLPASHAHLRVTAAPQSVCLARAVDQSVLLMKPDAELASSSVYNLLPEKDLTGFPGLNDQDDE
10 DCINRHNVIINGITYTPVSSTNEKDMYSFLEDMLKAFNTSKIRPKMCPQLQQYEMHGPEGLRVGFYSDVMGRGHARL
11 VHVVEPHTEVRKYFPETWIWDLVVVNSAGVAEAGVTVPDITTEWKAAGFCLSEDAGLISSTASLRAFQPFVELTMPY
12 SVIRGEAFTLKATVLNYPKICIRVSVQLEASPAFLAVPVEKEQAPHICICANGRQTVSWAVTPKSLGNVNFVSAEALQS
13 ELCGTEVPSVPEHGRKDTVIKPLLVEPEGLEKETTNSLLCPSGGEVSEELSLKPPNVVEESARASVSVLGDILGSAWQ
14 NTQNLQMPYGCGEQNMVLFAPNIYVLDYLNQTLQTPETIKSKAIGYLNTGYQRQLNYKHYDGSYSTFGERYGRNQNTW
15 LTAFLVLTFAQARAYIFIDEAHITQALIWLSQRQKDNQCFRSGSLNNNAIKGGVEDEVTLSAYITIIALLEIPLTVTHPV
16 VRNALFCLESAAWTAQEGDHGSHVYTKALLAYAFALAGNQDKRKEVLKSLNEEAVKKNDSVHWERPQKPKAPVGHFYPEQ
17 APSAEVMTSYVLLAYLTAPAPTSEDLSATNIWKWITKQQAQGGFSSTQDTVVALHALSKYGAATFRTGKAAQVTI
18 QSSGTFSKQVQVNNRLLQQQVSLPELPGEYSMKVTGEGCVYLQTSKYNI LPEKEEFPFALGVQTLPTQCDPEKAHTS
19 FQISLSVSYTGSRSASNMALVDVKKMVSGFIPKPTVKMLERSNHVSRTEVSSNHVLIYLDKVSNTLSLFFTVLQDVPVR
20 DLKPAIVKVDYDETDEFAIAEYNAPCSKDLGNA
21 >NP_000006.2 arylamine N-acetyltransferase 2 [Homo sapiens]
22 MDIEAYFERIGYKNSRNKLDLETLDILEHQRVAVPFENLMHCGQAMELGLEATFDHIVRRNRGGWCQVQVLLYWALT
23 TIGFQTTMLGGYFYIPPVNKYSTGMVHLLQVTDGRNYIVDAGSGSSSQMMQPLELISGKDQPVQPCIFCLTEERGIWY
24 LDQIRREQYITNKEFLNSHLLPKKHQKIYLFLEPRITIEDFESMNTYLOTSPTSSFITTSFCSLQTPGYVCLVGFIIT
25 YRKFVQNTDLVEFKLTTEEEVEEVLRNIFKISLGRNLVPKPGDGLTI
26 >NP_000007.1 medium-chain specific acyl-CoA dehydrogenase, mitochondrial isoform a precursor [Homo sapiens]
27 MAAGFGRCCRVLRSISRFWRSQHTKANRQREPGLGFSFEFTEQQKEFQATARKFAREEIIPVAAEYDKTGEYPVPLIRR
28 AWEGLMNHIPENCGGLGLTFDACLISEELAYGCTGVQTAIEGNSLGQMPITIIAGNDQKKKYLGRMTEEPLMAYCV
29 TEPGAGSDVAGIKTKAEKKGDEYIINGQKMWITNGGKANWYFLLARSDDPKAPANKAFTGFIVEADTPGIQIGRKE LNM
30 GQRCSDRGIVFEDVKVPKENVLIGDGAGFKVAMGAFDKTRPVVAAGAVGLAQRALDEATKYALERKTFGKLLVEHQAIIS
31 FMLAEAMAKVE LARMSYQRAAWEVDSGRNRTYYASIAKAFAGDIANQLATDAVQILGGNGFNTEYVPEKLMRDAKIYQIY
32 EGTSIQIRLIVAREHIDKYKN
33 >NP_000008.1 short-chain specific acyl-CoA dehydrogenase, mitochondrial isoform 1 precursor [Homo sapiens]
34 MAAALLARASGPARRALCPRAWRLHTIYQSVLEPETHQMLLQTCRDFAEKELFPIAAQVDKEHLFPAAQVKKMGGLGLL
```

Fonte: Elaborada pelo autor

Na Figura 1, podemos ver a padronização, com a separação dos cabeçalhos, facilitando a manipulação e extração de dados, que foi um dos focos deste trabalho.

Além disso, há uma grande facilidade de se obter estes arquivos em vários *websites*, com as mais variadas informações. O NCBI (*National Center for Biotechnology Information* – Centro Nacional para Informação Biotecnológica) é um deles, e foi utilizado para se obter arquivos *fasta* para este trabalho.

Figura 2 – Informações sobre a mosca do mediterrâneo

The screenshot displays the NCBI Genome browser interface for *Ceratitis capitata* (Mediterranean fruit fly). The top navigation bar includes links for Resources, How To, and Sign in to NCBI. The main search bar contains the query 'txid7213[orgn]' with a 'Search' button. Below the search bar, the organism overview is shown, including a description of the fly as a major agricultural pest and its taxonomic lineage. The summary section provides details about the genome assembly, including the submitter (The i5k Initiative), assembly level (Scaffold), and statistics (total length 436.491 Mb, protein count 22949, GC% 35.0995). The search details section shows the search term 'txid7213[orgn]'.

Fonte: Website do NCBI¹

Na Figura 2, podemos ver que há uma gama diversa de arquivos disponíveis para serem observados. Para este trabalho, foi utilizado apenas a sequência no formato *fasta* do genoma codificado para proteína. No entanto, também foi feita a extração de informações de arquivos oriundos do DISOPRED3, o qual se trata de um preditor de IDRs que utiliza o arquivo BLAST, que também pode ser obtido no NCBI.

Porém, a bioinformática não é apenas a extração e visualização de informações. Para ser melhor aproveitados, os dados são obtidos e utilizados para os mais variados fins, e através de vários métodos, seja utilizando *softwares* já existentes, ou com *scripts* desenvolvidos com funções específicas. Para o desenvolvimento destes códigos específicos, algumas linguagens de programação

¹ Disponível em: <[https://www.ncbi.nlm.nih.gov/genome/?term=txid7213\[orgn\]](https://www.ncbi.nlm.nih.gov/genome/?term=txid7213[orgn])>. Acesso em ago. 2019

são mais utilizadas do que outras, sendo as mais comuns *Python* e *Perl*. Os *scripts* deste trabalho foram desenvolvidos em *Python*, pois é uma linguagem de fácil aprendizado, já existindo conhecimento prévio por parte do autor, além de possuir uma grande comunidade até mesmo na área da bioinformática.

3 MATERIAIS E MÉTODOS

Aqui serão descritas as metodologias utilizadas para o desenvolvimento de todo o projeto, desde suas partes mínimas como o *script* para extrair informações do resultado do DISOPRED3 até a descrição das ferramentas para a construção do sistema *web*. Para este trabalho, foi utilizado um notebook Dell Inspiron 5470, possuindo um processador Intel Core i7-4510U com clock de 2.6GHz, uma memória RAM de 8GB e o Sistema Operacional Windows 10.

Para praticamente todo o trabalho, a linguagem de programação *Python* foi utilizada na sua versão 3.7.0. Além disso, outra ferramenta utilizada durante todo o trabalho foi o editor de texto Sublime Text 2, onde foram utilizados alguns pacotes para facilitar o trabalho.

3.1 Python/Sublime Text 2

O *Python* é uma linguagem de programação, como dito anteriormente, com grande auxílio na bioinformática. Contudo, apesar de existir conhecimento prévio por parte do auto em alguns módulos de bioinformática, em sua maioria, estes são voltados para a análise genômica, sem o mesmo estar codificado para proteínas. No entanto, não foi necessária a utilização de módulos para a manipulação local de arquivos, exceto para tratar expressões regulares.

Esta linguagem de programação trabalha bem com arquivos, possuindo funções práticas nesse sentido. Além disso, a simples integração com o Sublime Text 2 facilitou mais ainda o trabalho, possuindo interpretador embutido e auxiliado com a implementação de pacotes para funções diversas, tais como *Linter* (avaliar se o arquivo está nos padrões corretos), *Snippets* (preditores de código), realçadores de sintaxe ou o pacote *SublimeGit*, que faz a conexão com o *Git/GitHub* prática e com uma usabilidade otimizada.

Figura 3 – Ambiente de desenvolvimento

```

OPEN FILES
- split_proteins_local.py
- find_entries_diso.py
FOLDERS
- tcc
  - files-tcc
  - src
  - __pycache__
  - not_splitted_files
  - templates
    - app.py
    - find_entries_diso.py
    - GCF_000195955.2_ASM19595v2_protein.txt
    - GCF_000347755.3_Ccap_2.1_protein_2.txt
    - run-fed.py
    - sender.py
    - split_proteins.py
    - split_proteins_local.py
    - Ta04_gD18130.diso
    - Ta04_gD18130.txt
  - tests
    - .gitignore
    - README.md
    - tcc.sublime-project
    - tcc.sublime-workspace
split_proteins_local.py
1 # Script para dividir o arquivo 'fasta', de forma a igualar o proces-
2 # samento entre os núcleos (max de 20)
3
4 # Nome do arquivo a ser processado abaixo, associado a esta variável
5
6 proteins_fasta = 'GCF_002204515.2_AaegL5.0_protein'
7
8 new_file = open(proteins_fasta + '.txt', 'w')
9
10 # Numero de divisões para a proteína ser processada
11
12 num_divisions = 3
13
14 # Função para ler um arquivo, recebendo o nome do arquivo como entrada
15
16 def read_file(filename):
17
18     fasta_file = open(filename + '.faa')
19
20     return fasta_file
21
22
23 # Função que recebe um arquivo como entrada, e retorna uma lista com
24 # os cabeçalhos e proteínas
25
26 def saveSplitted(fasta_file):
27
28     # Declaração de uma lista vazia, onde vão ser inseridos os cabeçalhos
29     # com sua respectiva sequência em seguida. Modelo de lista abaixo:
30     # ["cabeçalho_1", "sequencia1", "cabeçalho_2", "sequencia_2"] ...
31
32     splitted_elements = []
33
34
35 # Leitura do arquivo e atribuição de cada linha do arquivo a um elemen-
36 # to de uma lista
37
38 lines = fasta_file.readlines()
39

```

Fonte: Elaborada pelo autor

O *Python* mostrou ser uma linguagem de programação tão eficiente para ser desenvolvido o trabalho, que foi buscado um *framework web* nesta linguagem para o sistema *web* ser criado, e a melhor opção para a situação foi o *Flask*.

3.2 Flask

Como havia a necessidade, de alguma forma, do *script* executar em documentos submetidos na *web*, foi feita uma análise do melhor *framework web* para tratar arquivos com *Python*. A opção de utilizar o *Django*, *framework web* criado através do *Python* mais famoso, foi descartada, pois o mesmo é voltado para o desenvolvimento de *websites* em geral, sem haver necessariamente a existência de programas da linguagem embutidos nele.

Figura 4 – Flask e Python

Fonte: Kenya Tech²

Para o aprendizado da utilização do *framework*, no entanto, houve a interferência do *Django*, por ter mais informação acessível, como cursos e tutoriais. A forma de utilização do *Flask* foi através de um ambiente virtual (*virtualenv*), característica oferecida pelo *Python*, onde é feito um projeto local dedicado em que é simulado um servidor, podendo assim fazer os testes no navegador.

Figura 5 – Server rodando através do *Flask*

```
C:\Windows\system32\cmd.exe - python sender.py

C:\Users\icaro\Documents\tcc\tests\flask-project\flask-project>Scripts\activate.bat

(flask-project) C:\Users\icaro\Documents\tcc\tests\flask-project\flask-project>cd C:\Users\icaro\Documents\tcc\src

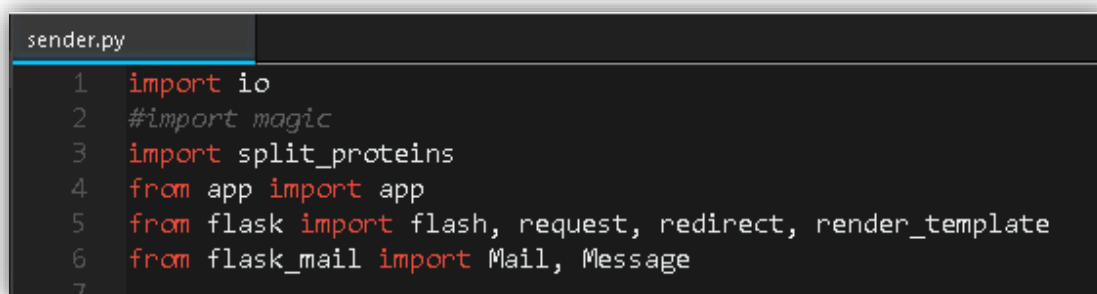
(flask-project) C:\Users\icaro\Documents\tcc\src>python sender.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Fonte: Elaborada pelo autor

² Disponível em: < <https://kenya-tech.com/2019/01/14/getting-started-with-flask-python/>>. Acesso em ago. 2019

O *Flask*, na contramão do *Python*, é categorizado como um *microframework*, ou seja, é um *framework* que necessita de dependências e os chamados *imports* para praticamente tudo além do “*hello world*” que se desejar fazer. No entanto, é possível fazer bastante coisa com estes módulos, o que acaba não atrapalhando em nada o fato do *Flask* ter essa característica.

Figura 6 – *Imports* necessários para rodar o servidor *web*

A screenshot of a code editor with a dark background. The file name 'sender.py' is visible in the top left corner. The code consists of seven lines of Python imports, with line numbers 1 through 7 on the left. The imports are: 1. 'import io', 2. '#import magic' (commented out), 3. 'import split_proteins', 4. 'from app import app', 5. 'from flask import flash, request, redirect, render_template', 6. 'from flask_mail import Mail, Message', and 7. (empty line).

```
sender.py
1  import io
2  #import magic
3  import split_proteins
4  from app import app
5  from flask import flash, request, redirect, render_template
6  from flask_mail import Mail, Message
7
```

Fonte: Elaborada pelo autor

Apesar do desenvolvimento parecer estranho à primeira vista, após um tempo, o *framework* tornou-se intuitivo e amigável, permitindo transportar as funções feitas localmente com a intenção de manipular os arquivos para uma interface *web*. No entanto, o *Flask* não poderia resolver todos os problemas sozinho. Embora tenha facilitado o trabalho, ele não pode disponibilizar as informações por si só, precisando do trabalho em conjunto com outras linguagens.

3.3 HTML, CSS & Bootstrap

Para representar os dados na *web*, foi criada uma página simples, porém intuitiva, através das linguagens já conhecidas popularmente para desenvolvimento *web* (HTML e CSS), além do auxílio do *framework Bootstrap* para simplificar o trabalho, adicionando elementos pré-modelados.

Além disso, o próprio *Flask* trabalha com trechos e *features* do HTML, como redirecionamento de URL, e até mesmo renderização de páginas inteiras, seja no próprio arquivo, ou com chamadas de outros arquivos. No caso deste trabalho, o

arquivo principal do *Flask* chama o *template* desenvolvido em HTML/CSS/Bootstrap, e passa ou recebe as informações necessárias, para depois processá-las.

3.4 Outros *softwares*

Alguns outros *softwares* não foram utilizados diretamente, contanto, devem ser citados, pois os arquivos obtidos como resultado do processamento destes programas foram utilizados.

O principal deles foi o DISOPRED3, que se trata de um *software* onde é submetido um arquivo no formato *blast*, e é retornado um arquivo no formato *diso*. Este arquivo retornado foi utilizado no projeto, sendo desenvolvido um *script* em cima deste modelo.

Há, no entanto, um sistema que foi utilizado diretamente, porém sem relação direta com desenvolvimento (poderia ser concluído o projeto sem sua utilização): o GitHub. Trata-se de um Sistema de Controle de Versão, onde é possível hospedar os arquivos de um determinado projeto, e seu estado, para, em caso de perda, ser recuperado de forma simples e prática.

3.5 Funcionamento do projeto

O método de desenvolvimento do projeto foi separado em 3 etapas:

3.5.1 Desenvolvimento local

A primeira parte do projeto consistiu em realizar o desenvolvimento dos *scripts* para que eles funcionassem localmente. O desenvolvimento dos *scripts* poderia e foi feito em paralelo, pois não havia interdependência entre eles. Um reorganiza dados extraídos do resultado de predição de IDRs no DISOPRED3, enquanto o outro divide o arquivo *fasta* de acordo com seu número de aminoácidos em uma quantidade pré-determinada, para que ele seja processado pelo supercomputador CACAU da UESC.

Porém, o primeiro foi finalizado apenas na fase de *script*, enquanto o segundo foi levado à próxima etapa. Nesta primeira etapa, a linguagem de programação *Python* foi utilizada, assim como um arquivo *diso* para serem executados os testes do *script*. Além disso, nenhum *software* externo foi utilizado

Figura 8 – Arquivo *sender.py*, responsável por administrar o *website*

```

if file.filename == '':
    flash('Selecione um arquivo!')
    return redirect(request.url)
if file and final_words in ALLOWED_EXTENSIONS:
    mail_subject = ("Nova submissão - Arquivo fasta - " +
                    file.filename
                    )

    mail_body = (
        "Arquivo .fasta submetido por " +
        name +
        "\nEmail: " +
        email +
        "\nNúmero de clusters solicitado: " +
        str(number)
    )

    msg = Message(
        subject=mail_subject,
        sender=app.config.get("MAIL_USERNAME"),
        recipients=[app.config.get("MAIL_USERNAME")],
        body=mail_body
    )

    wrapper = io.TextIOWrapper(file)

    splitted_files = split_proteins.names(raw_name, wrapper, number)]

```

Fonte: Elaborada pelo autor

Como é possível ver na Figura 8, há uma mescla de informações nesta etapa: enquanto a maior parte do código trata-se de *Flask*, alguns trechos recorrem ao *script* desenvolvido localmente (*split_proteins*).

3.5.3 Desenvolvimento Web

Por fim, houve o desenvolvimento da página em si, com algumas validações de segurança e estilização, pois os dados já foram trabalhados e direcionados para serem enviados pelo *Flask*.

4 RESULTADOS E DISCUSSÕES

Ao final, foram concluídos dois resultados: primeiramente, um *script* simples, porém eficiente, chamado *find_entries_diso.py*, onde é dado como entrada o nome de um arquivo *diso* presente na mesma pasta do *script*.

Este arquivo *diso* é formatado contendo uma lista de todos os aminoácidos, um por linha, e, em cada uma desta, é apresentado se é uma IDR através de um asterisco (*), ou se é uma região ordenada através de um ponto (.).

Este *script*, então, percorre todo o arquivo, buscando pelas IDRs, onde:

- Abaixo de três aminoácidos não é considerada uma Região Intrinsecamente Desordenada;
- Se existir uma região entre 3 e 30 aminoácidos desordenados em sequência, esta região é considerada uma IDR curta
- Qualquer região acima de 30 aminoácidos desordenados em sequência é considerada uma Região Intrinsecamente Desordenada longa.

Como resultado, ele emite um arquivo *txt*, onde mostra as regiões desordenadas (seus inícios e fins), além de mostrar o número total de IDRs curtas e longas.

Tabela 2 – Formato do arquivo *diso*

Letra do aminoácido	Posição do aminoácido	Flag
R	25	*
L	26	*
L	27	*
G	28	*
F	29	.
M	30	.
F	31	*
G	32	*
N	33	*

Fonte: Elaborado pelo autor

Figura 9 – Arquivo *txt* obtido com a execução do *script*

```

Tc04_g018130.txt
1  ['1'] - ['28'] = 28
2  ['31'] - ['33'] = 3
3  ['35'] - ['243'] = 209
4  ['268'] - ['371'] = 104
5  ['390'] - ['393'] = 4
6  ['395'] - ['508'] = 114
7  ['527'] - ['529'] = 3
8  ['556'] - ['623'] = 68
9  ['633'] - ['645'] = 13
10 ['662'] - ['693'] = 32
11 ['705'] - ['724'] = 20
12 ['742'] - ['759'] = 18
13 ['792'] - ['826'] = 35
14 ['849'] - ['860'] = 12
15 ['871'] - ['881'] = 11
16 ['904'] - ['909'] = 6
17 ['960'] - ['969'] = 10
18 ['1061'] - ['1065'] = 5
19 ['1082'] - ['1108'] = 27
20 ['1145'] - ['1151'] = 7
21 ['1155'] - ['1393'] = 239
22 ['1411'] - ['1413'] = 3
23 ['1419'] - ['1479'] = 61
24 ['1499'] - ['1827'] = 329
25 ['1940'] - ['1949'] = 10
26
27 Long residues = 16
28 Short residues = 9

```

Fonte: Elaborado pelo autor

Além deste, foi desenvolvido um *website* onde, de forma integrada, manipula arquivos. Trata-se de um formulário onde devem ser preenchidos o nome, e-mail (para fins de contato), o número de clusters a serem utilizados no CACAU (conforme indicação prévia) e um campo para enviar um arquivo.

Ao ser enviado, o arquivo automaticamente é dividido de acordo com seu número total de aminoácidos, deixando a quantidade de elementos em cada arquivo individual a mais equilibrada possível. Alguns testes foram feitos para avaliar a diferença entre cada parte. A Tabela 3 foi feita em cima do arquivo *fasta* do genoma da mosca do Mediterrâneo:

Tabela 3 – Variação percentual entre a menor e maior parte (tamanho em quantidade de aminoácidos)

Menor parte	Maior parte	Número de divisões	Variação Percentual (arredondado)
5807193	5813872	3	11%
4356206	4359457	4	7%

3485555	3486699	5	3%
2901791	2908381	6	2%
2488719	2491408	7	10%

Fonte: Elaborado pelo autor

Este arquivo é então enviado para um endereço de e-mail (disopred.uesc@gmail.com), onde pode ser utilizado por alguém que gerencie o uso do CACAU, e possa submeter os arquivos para serem processados de acordo com os núcleos disponíveis.

Figura 10 – *website* onde o arquivo é submetido para ser enviado via *e-mail*

Sistema de análise de proteínas .fasta

Insira seu nome aqui:

Insira o número de clusters a serem utilizados:

Endereço de email

Selecione o arquivo no formato fasta ('faa', 'fasta', 'fna', 'ffn', 'frn', 'multifasta' ou 'multi-fasta'):

Nenhum arquivo selecionado

Fonte: Elaborado pelo autor.

Este *website* estará disponível no endereço <http://nbcgib.uesc.br/disopred>.

5 CONSIDERAÇÕES FINAIS

Foi possível constatar, com este trabalho, a importância do estudo das proteínas, pois esta é a macromolécula mais presente no dia-a-dia. Além disso, é perceptível que os estudos relativos às IDPs estão apenas começando, pois há pouquíssima disponibilidade de informação nesta área, bem como *softwares* que trabalhem com estes tipos de dados, sendo muitas vezes necessário o que foi feito aqui, ou seja, o desenvolvimento de *scripts* próprios para a execução do trabalho desejado.

Além disso, torna-se evidente a predominância dos *frameworks* no desenvolvimento de projetos, pois, além de agilizarem o trabalho trazendo coisas complexas já quase “prontas” e simplificar os trabalhos medianos, alguns *frameworks* (como o *Flask*) trazem recursos que nem mesmo as linguagens de desenvolvimento *web* possuíam, ou possuem com uma grande quantidade de exigências.

6 TRABALHOS FUTUROS

Há alguns trabalhos que podem ser desenvolvidos a partir deste projeto:

- a) Integração do *script* de análise do arquivo *diso* com o resultado de outro programa, como por exemplo o InterPro Scan, onde este emite alguns relatórios contendo as funções das proteínas. Unindo as informações de ambos torna fácil o comparativo;
- b) Criar um sistema com cliente/administrador para o *website*, onde apenas pessoas autorizadas terão acesso e poderão submeter arquivos;
- c) Analisar a possibilidade de submeter os arquivos diretamente para processamento no CACAU mediante autorização prévia;

REFERÊNCIAS

CELI, R. **Proteínas: o que é, função e fontes de proteína!** 2019. Disponível em: < <https://www.stoodi.com.br/blog/2019/03/13/proteinas-o-que-e/> >. Acesso em: 26/07/2019.

LI, J. et al. **An Overview of Predictors for Intrinsically Disordered Proteins over 2010–2014.** International Journal of Molecular Sciences, 2015. p. 23447.

MENG, F., UVERSKY, V., & KURGAN, L. **Computational prediction of intrinsic disorder in proteins.** *Current Protocols in Protein Science*, 2017. 88,2.16.1.

PINHEIRO, P. **O QUE SÃO PROTEÍNAS E AMINOÁCIDOS?** 2019. Disponível em: < <https://www.mdsaude.com/nutricao/proteinas/> >. Acesso em: 30/07/2019.

National Center for Biotechnology Information. **Ceratitis capitata (Mediterranean fruit fly).** Disponível em < [https://www.ncbi.nlm.nih.gov/genome/?term=txid7213\[orgn\]](https://www.ncbi.nlm.nih.gov/genome/?term=txid7213[orgn]) >. Acesso em 17/08/2019

Pallets. **Flask Documentation.** Disponível em < <https://flask.palletsprojects.com/en/1.0.x/quickstart/> >. Acesso em 14/08/2019

OLIVEIRA, K. de F. S. **Conhecendo o Jinja2: um mecanismo para templates no Flask.** Disponível em < <https://imasters.com.br/desenvolvimento/conhecendo-o-jinja2-um-mecanismo-para-templates-no-flask> >. Acesso em 18/08/2019

ROBINSON, K. **Send Email programmatically with Gmail, Python, and Flask.** Disponível em < <https://www.twilio.com/blog/2018/03/send-email-programmatically-with-gmail-python-and-flask.html> >. Acesso em 18/08/2019

FÁTIMA H. **Flask x Django: como escolher o framework correto para seu aplicativo web.** Disponível em < <https://imasters.com.br/back-end/flask-x-django-como-escolher-o-framework-correto-para-seu-aplicativo-web> >. Acesso em 10/08/2019

Django Girls. **Seu primeiro projeto Django!** Disponível em < https://tutorial.djangogirls.org/pt/django_start_project/ >. Acesso em 10/08/2019

Pallets. **Werkzeug**. Disponível em < <https://werkzeug.palletsprojects.com/en/0.15.x/> >. Acesso em 13/08/2019

Pythonspot. **Flask web forms**. Disponível em < <https://pythonspot.com/flask-web-forms/> >. Acesso em 13/08/2019

OTTO, M. **Formulários em Bootstrap**. Disponível em < <https://getbootstrap.com.br/docs/4.1/components/forms/> >. Acesso em 10/08/2019

Creative Commons Atribuição 4.0 Internacional. **Python**. Disponível em < <http://www.devfuria.com.br/python/> >. Acesso em 10/08/2019

FILHO, M. C. **PythOn Flask — Iniciando ao Framework**. Disponível em < <https://medium.com/@cfmedeiros/python-flask-iniciando-ao-framework-efd66250add4> >. Acesso em 10/08/2019

ROCHA, B. C. **What the Flask? Pt-1 Introdução ao desenvolvimento web com Python**. Disponível em < <http://pythonclub.com.br/what-the-flask-pt-1-introducao-ao-desenvolvimento-web-com-python.html> >. Acesso em 13/08/2019

MDBootstrap.com. **Bootstrap File Input**. Disponível em < <https://mdbootstrap.com/docs/jquery/forms/file-input/> >. Acesso em 13/08/2019

MOHIT. **Must-Have Packages and Settings in Sublime Text for a Python Developer**. Disponível em < <https://fosstack.com/setup-sublime-python/> >. Acesso em 05/08/2019

Python Software Foundation. **HOWTO Use Python in the web**. Disponível em < <https://docs.python.org/2/howto/webrowsers.html> > Acesso em 05/08/2019.

FREITAS, V. **How to Upload Files With Django.** Disponível em <
<https://simpleisbetterthancomplex.com/tutorial/2016/08/01/how-to-upload-files-with-django.html> > Acesso em 05/08/2019.

PythonAnywhere. **A beginner's guide to building a simple database-backed Flask website on PythonAnywhere.** Disponível em <
<https://blog.pythonanywhere.com/121/> > Acesso em 12/08/2019.

Real Python. **Python-driven Web Applications.** Disponível em <
<https://realpython.com/python-web-applications/> > Acesso em 12/08/2019.