Trabalho 2 de Algoritmos de Bioinformática: Alinhamento Pareado de Sequências

Edgar Carneiro
Faculdade de Ciências da Universidade do Porto
(Dated: 14 de Abril de 2019)

I. INTRODUÇÃO

No âmbito da disciplina de Bioinformática, disciplina esta que aborda temáticas atuais e relevantes na área da investigação, será analisada a implementação de um módulo para gerir o alinhamento pareado de sequências.

O alinhamento pareado de sequências tem como objetivo organizar sequências de tal forma a que o nível de semelhança entre estas seja máximo. O módulo desenvolvido tem como foco situações nas quais existem múltiplas formas de obter um alinhamento ótimo entre as sequências.

Esta implementação trata-se de uma base para futuros trabalhos desenvolvidos na disciplina previamente mencionada.

II. DESCRIÇÃO E ESTRATÉGIAS DE IMPLEMENTAÇÃO

O maior problem encontrado no desenvolvimento deste projeto foi a adaptação das funções de recuperação de alinhamentos globais e locais, sendo que as restantes funcionalidades foram desenvolvidas sem que relevantes decisões estruturais ou de implementação emergissem.

Foram consideradas duas formas de implementação para as funções de recuperação de alinhamento, quer global quer local:

1. **implementação iterativa:** nesta implementação é mantida uma lista de possíveis alinhamentos a percorrer, sendo que quando na matriz *Trace* de um alinhamento existirem mais do que uma hipótese de procedimento, as várias opções são adicionadas à lista de possíveis alinhamentos a percorrer. O algoritmo termina quando não existirem mais possíveis alinhamentos a percorrer, sendo que essa situação só é alcançada quando todos os elementos da lista tiverem chegado à posição [0, 0] na sua matriz *Trace*. Esta implementação segue a estrutura sugerida pela Pseudo-código fornecido pelo professor.

2. **implementação recursiva com *memoization*:** esta implementação faz uso da recursividade que se adapta ao problema em mão. Nesta abordagem, sempre que na matriz *trace* forem encontradas várias opções de procedimento, serão desencadeadas as respetivas chamadas recursivas sendo depois o resultado destas recolhido numa lista, e assim recursivamente. No entanto, esta opção não seria viável sem o uso de ***memoization*** pois teria complexidade temporal e espacial maior que a implementação iterativa. Com *memoization*, em cada passo da recursão, é mantida uma matriz atualizada com os resultados de recursões prévias, evitando assim que recursões anteriormente realizadas se tenham de realizar de novo. Como visto em [2], recursão com *memoization* é mais rápida em execução do que a implementação iterativa.

Pela sua vantagem a nível temporal, foi usada a **implementação recursiva com *memoization***. Podemos ainda ver um exemplo da vantagem desta implementação na figura 1. Nesta figura, a primeira matriz representa a *memoization matrix* no inicio do algoritmo. A segunda matriz representa a *memoization matrix* no final da recursão para um primeiro alinhamento local. A terceira representa a *memoization matrix* no início do algoritmo para um outro alinhamento local. Como é possível observar, apenas é necessário calcular uma iteração visto as restantes já se encontrarem calculadas na *memoization matrix*, sendo essa a diferença da terceira matriz para a quarta matriz apresentada.

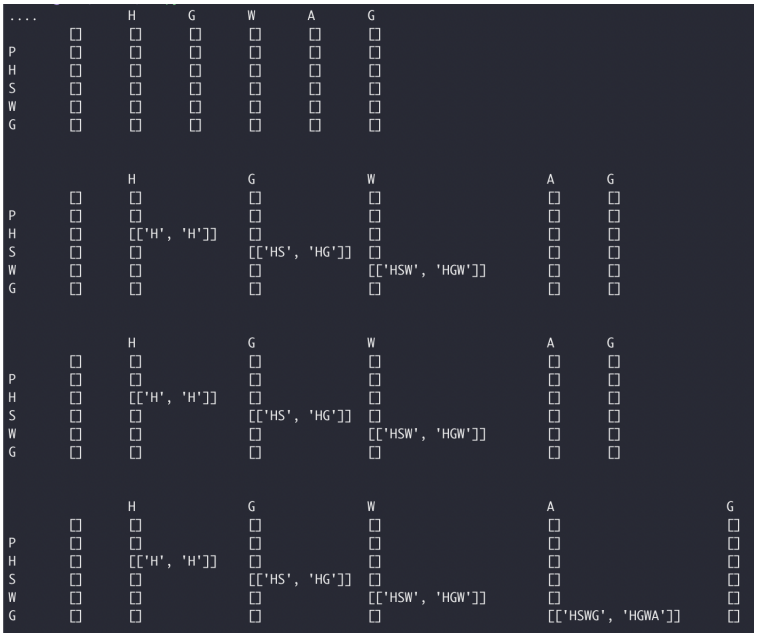

Figura 1. Exemplo de vantagem de *Memoization*

III. RESULTADOS

Foram implementadas as funcionalidades pedidas na sua totalidade. Todas as funções implementadas encontram-se com assinaturas semelhantes às requisitados no documento de apresentação de problema.

De destacar que as funcionalidades da secção A), B) e D) no enunciado se encontram implementadas no módulo `SeqAlign.py`. Por outro lado a funcionalidade C) encontra-se implementada através da bateria de testes unitários desenvolvidos, bem como no programa `run_me.py` desenvolvido para demonstração das funcionalidades implementadas.

A. Funcionalidades Extra

Foram desenvolvidas várias funções extra, para facilitar o uso do módulo previamente referido, sendo estas:

- `read_submat_file:` função que retorna uma matriz de substituição presente no ficheiro passado como argumento.
- `subst_matrix:` função que cria um matriz de substituição com base num dado alfabeto e valores de *match* e *mismatch*.
- `pretty_matrix:` função que recebe uma matriz e as legendas em linha e coluna e faz o *output* desta de uma forma visualmente agradável. Um exemplo do *output* desta função pode ser visualizado na figura 1.

Foram também desenvolvidas duas funcionalidades extra, sendo estas:

- `compare_pairwise_num_global_align:` função que recebe uma lista de sequências e indica qual o número de possíveis alinhamentos globais existentes entre os possíveis pares de sequências pertencentes à lista dada.
- `compare_pairwise_num_local_align:` função que recebe uma lista de sequências e indica qual o número de possíveis alinhamentos locais existentes entre os possíveis pares de sequências pertencentes à lista dada.

Todas as funcionalidades foram ainda testadas atingindo perto de 95% de coverage (apenas os métodos *pretty_print()* não foram testados). Para confirmar os resultados obtidos foi também usada uma ferramenta desenvolvida pela *bioninformatics.org*[1].

Foi também realizada documentação da totalidade do código.

IV. COMENTÁRIOS E CONCLUSÕES

Acredito que o meu conhecimento sobre alinhamento pareado de sequências foi consideravelmente aprofundado. De facto, a área da bioinformática responsável pelo alinhamento de sequências é uma área onde a componente informática tem um forte impacto, sendo responsável pela execução de tarefas que seriam de outra forma exaustivas e complexas, contribuindo assim ainda mais para o interesse no desenvolvimento do projeto.

Em jeito de conclusão, o resultado obtido com este trabalho foi satisfatório, tendo implementado a totalidade das funcionalidades requisitadas. Considero também que o trabalho desenvolvido, pela estrutura que apresenta, revela-se uma base sólida para possíveis extensões e desenvolvimentos sobre esta.

V. BIBLIOGRAFIA

[1] Sequence Manipulation Suite:, http://www.bioinformatics.org/sms2/pairwise_align_dna.html

[2] Iteration and Recursion in Python, http://geodesygina.com/Fibo.html