Introdução à Ciência da Computação II SSC0503

Professor: Adenilso da Silva Simão
PAE: Jorge Francisco Cutigi
Monitor: Luis Felipe Jorge

Atividade avaliativa 11

Algoritmo Guloso

Data de entrega: 14/12/2020

Instruções:

Os trabalhos devem ser entregues no run.codes, na disciplina com código SHEL. A atividade será realizada em **grupo formado por 3 ou 4 alunos**. Todos os integrantes devem submeter o mesmo código. O grupo deve se "reunir" ao menos uma vez pelo google meet para discutir/resolver o problema e deve gravar a sessão. Além do código, **deve ser postado por apenas um integrante do grupo** o link para o vídeo da sessão gravada (no máximo 5 minutos) no fórum específico no e-disciplinas, **juntamente com a indicação dos integrantes do grupo**. O vídeo deve estar aberto para visualização de todos os alunos (compartilhar como "Qualquer pessoa da USP pode visualizar").

Atividade:

De uma maneira bastante simplificada, pode-se dizer que o sequenciamento do DNA é realizado a partir da leitura de vários fragmentos (chamadas de *reads*), que são “montadas” posteriormente.
Nessa atividade vamos simular a montagem do DNA a partir das *reads* sequenciadas. Para isso, faça um programa em C que leia um número inteiro *N* que indica a quantidade de *reads*, seguido da leitura de cada uma das *reads*. Após isso, utilize algoritmo guloso para montar essas *reads*, gerando como resultado apenas uma sequência. A estratégia do algoritmo guloso deve considerar a melhor sobreposição (*overlap*) entre as reads em cada etapa.

Exemplo:
Considere o conjunto seguinte de reads:

- READS = [`atccat, ctgatc, ccatg`]

Primeira iteração das permutções das *reads* e possíveis montagens:

- `atccat + ctgatc -> atccatctgatc -> overlap 0`
- `atccat + ccatg -> atccatg -> overlap 4`
- `ctgatc + atccat -> ctgatccat -> overlap 3`

- `ctgatc + ccatg -> ctgatccatg -> overlap 1`
- `ccatg + atccat -> ccatgatccat -> overlap 0`
- `ccatg + ctgatc -> ccatgctgatc -> overlap 0`

Portanto, o maior overlap é de tamanho 4. Remove-se as sequências envolvidas nesta montagem e insere a nova sequência montada na primeira posição da lista de reads, tendo como resultado:

- READS = [`atccatg`, `ctgatc`]

Repete-se o procedimento com a nova lista gerada

Observações importantes:

1. As permutações devem seguir a ordem ilustrada. Por exemplo, em um vetor v = [0, 1, 2, 3], as permutações devem ser na seguinte ordem: (0, 1), (0, 2), (0, 3), (1, 0), (1, 2), (1, 3), (2, 0), (2, 1), (2, 3), (3, 0), (3, 1), (3, 2).
2. Se um mesmo *overlap* for identificado, considerar a primeira encontrada.
3. Caso não aconteça sobreposição (overlap = 0), deve-se realizar a montagem das reads mesmo assim, concatenando as reads sem sobreposição.
4. Sobreposições completas no interior das *reads* também devem ser consideradas. Por exemplo, a combinação das reads “tcg” e “actcgaac” tem *overlap* 3 e a sequência resultante é a própria sequência “actcgaac”, pois “tcg” é substring de “actcgaac”.

Exemplo de entrada e saída:

Entrada	Saída
3 atccat ctgatc ccatg	ctgatccatg