Universidade de São Paulo
Instituto de Matemática e Estatística

MAC0323 - Algoritmos e Estrutura de Dados
2023

Exercício-programa 3: Reconstrução de DNA's

Beatriz Viana Costa

Conteúdo

1. Algoritmo e escolhas de implementação

Para a realização do atual exercício programa e a reconstrução do DNA, os principais algoritmos necessários foram a leitura e construção do grafo, posteriormente a retirada de quaisquer ciclos nele presentes, e após isso a execução de sua ordenação topológica.

Para a leitura do grafo foram realizadas comparações da sequência do novo nó a ser inserido com os já presentes no grafo, fazendo as conexões, se houverem.

Para a realização da retirada dos ciclos foi necessário usar o algoritmo de busca em profundidade recursiva (*dfsR*), foram utilizados vetores de booleanos *marked* e *onStack* que marcavam se o vértice atual já havia sido visitado pela função de dfs e se o vetor já havia sido visitado pela função de dfs naquela rodada, respectivamente. No caso do vetor *marked*, ele é marcado como *true* apenas uma vez, e seu valor nunca é alterado, diferente do vetor *onStack* que é marcado como *true* quando entra no dfs, porém é desmarcado ao sair.

A cada iteração do do dfs recursivo, há uma verificação se o vértice visitado não está marcado como verdadeiro no vetor *marked*, caso esteja, significa que a aresta observada faz com que se forme um ciclo, então a ligação é excluída.

Finalmente para a realização da ordenação topológica, é executado um algoritmo novamente um dfs recursivo, com o mesmo vetor *marked*, contudo, agora contamos o tempo em que cada vértice termina de ser observado (ou seja, todos os vértices adjacentes à ele também já acabaram de ser visitados).

Dessa forma é construído um vetor com os tempos em que cada vértice foi finalizado. Após isso os tempos são ordenados de forma decrescente, o que nos traria um caminho máximo no grafo. Assim, ao fim as sequências de cada vértice são impressas na tela, retirando-se a intersecção entre cada uma.

2. Execução e testes

Para a compilação do programa basta digitar o seguinte comando no terminal:

$ make

Já para a execução do programa:

$ ./ep3

Serão pedidos o nome do arquivo de entrada, que deverá ser indicado com a extensão *.txt* e, posteriormente, um valor de k, que será a quantidade mínima de letras iguais no fim e início de cada palavra para que haja uma aresta ligando um fragmento a outro. Após isso os dados serão processados e o programa printa o resultado encontrando do que seria uma reconstrução do DNA.

Os arquivos utilizados no momentos dos testes estão inclusos no arquivo *zip* entregue.

Seguem os resultados dos seguintes testes:

Algoritmo 1: Exemplo 1 com $k = 2$

```
1  12
2  ACTCGT
3  ATACATAA
4  TAACGA
5  ACAGAT
6  TCGTA
7  AAATA
8  ATAAC
9  CGAT
10 GTAAATA
11 ACATAA
12 GATAC
13 GATAC
14
15 ACTCGTAAATACATAACGATAC                    // Sequência original de DNA
16                                                  // Fim da entrada
17 ACTCGTAAATAACGATACATAAATAACAGATACATAA                      // Saída
```

Algoritmo 2: Exemplo 2 com $k = 2$

```
14
ATGGATCGATAGCTAGTAAC
AGGCAGTGAA
TGAATTACCCGATCTAGCTAGCATGGATCGATA
GTCACG
CGATCTAG
ATCGATAGCTAGTA
AGTCACGTC
GATCGATAGCTAG
CTAGTAAC
TCACGTCAGGCAGTGA
AGTAAC
GTCACGTCAGGCAG
GATCGATAGC
ATAGCTAGTAACA

AGTCACGTCAGGCAGTGAATTACCCGATCTAGCTAGCATGGATCGATAGCTAGTAACA // Sequência original de DNA
                                                              // Fim da entrada
GTCACGATCTAGGCAGTGAATTACCCGATCTAGCTAGCATGGATCGATAGCTAGTCACG
TCAGGCAGTGATCGATAGCTAGTATGGATCGATAGCCTAGTAAC                  // Saída
```

Algoritmo 3: Exemplo 3 com $k = 2$

```
10
ATTTCAGTTCGATAGCA
TGGATCTA
CTATCGATATC
CTGGATCT
TAGCTATCGAT
GCTATCGAT
TCGATATCAGC
CGATAGCA
ACTGGA
AGCGATTTCAGTT

ACTGGATCTATTGCTAGCTATCGATATCAGCGATTTCAGTTCGATAGCA // Sequência original de DNA
                                                  // Fim da entrada
ACTGGATCTAGCTATCGATATCAGCGATTTCAGTTGCTATCGATTTCAGTTCGATAGCA // Saída
```

3. Conclusão

Com os testes realizados foi possível notar que as sentenças devolvidas pelo programa condizem em maior parte com a sentença original, trazendo resultados satisfatórios, uma vez que este é um problema difícil de se resolver e os algoritmos implementados não foram tão refinados.

Além disso, em outros testes realizados com os mesmos arquivos de entrada, contudo com valores de k maiores não trouxeram resultados finais melhores, na verdade, foi exatamente o contrário, para os testes realizados com $k = 2$ devolveu os fragmentos reconstruídos com a maior semelhança com a sequência de DNA original.

Por fim é importante ressaltar que a solução final, com a utilização dos algoritmos aqui apresentados, pode não ser única e também não ser a melhor, uma vez que o resultado final depende das arestas a serem removidas na hora de eliminar os ciclos do grafo e também por qual vértice começamos na ordenação topológica.