**1. Is the graph from Fig 8.22 Eulerian (in other words: what’s the result of your function is\_eulerian, given this graph as input)?**

*Yes, this graph is eulerian as it contains a eulerian cycle.*

*The path found is: A->B->C->J->F->E->J->D->C->I->H->F->G->A*

**2. Does the graph from Fig 8.22 contain a Eulerian path?**

*A graph contains a Eulerian path if at most 2 nodes are semi-balanced and the other nodes are balanced. In this case all nodes are balanced so the graph does contain a Eulerian path .*

**3. Print the Eulerian path that your code can find in graph\_822.**

**4. If you run it 3 times (print your results below), do you always find the same path? Why or why not?**

Using the way this is implemented in my code, the eulerian path is in this case identical to the cycle, because all nodes are balanced. A→B→C→J→F→E→J→D→C→I→H→F→G→A. This stays the same if the script is ran multiple times.

**5. Print the graph that you constructed from the spectrum s (Fig 8.20). Use: for k, v in graph.items(): print k, v**

*The graph based on the spectrum:*

*('GT', ['TG'])*

*('CA', [])*

*('CG', ['GT'])*

*('GG', ['GC'])*

*('GC', ['CA', 'CG'])*

*('AT', ['TG'])*

*('TG', ['GG', 'GC'])*

**6. Is this graph Eulerian? Why or why not? And does it contain an Eulerian path? Why or why not?**

*This graph is not Eulerian, as not all nodes are balanced. CA is not pointing to anything but is pointed to. The same applies to AT but the other way around. The graph however does contain an Eulerian path as both CA and AT are semi-balanced and all other nodes are balanced.*

**7. Print the Eulerian path that your algorithm finds. To which DNA sequence does this path correspond? Print the sequence.**

T->G->G->C->C->G->G->T->T->G->G->G->G->C->C->A

**8. Which Eulerian cycle or path (if any) do you find in the bigger\_graph (provided in the skeleton)?**

*The path found does not seem to be correct:*

*1->2->3->9->4->5->6->7->10->11->4->3->1*

**9. Is the algorithm exact or approximate?**

*Exact, if there is an Eulerian cycle it will find it. If there is a Eulerian path it will find it.*

**10. Is the running time of Euler’s algorithm proportional to the number of nodes or the number of edges in the graph? Can you explain why?**

*The algorithm is proportional to the number of edges. The number of nodes is trivial in this sense because each node can have either none, one or more than one edges. Therefore the number of nodes is not directly traceable to the running time complexity.*