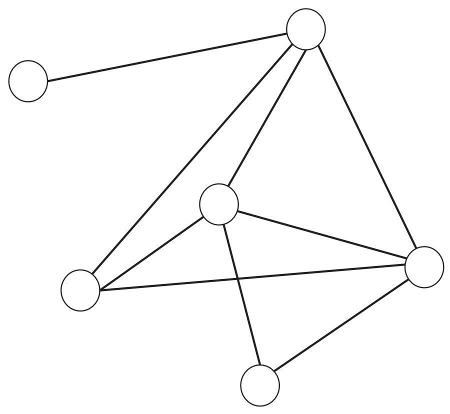
Algorithms for Bioinformatics (autumn 2010)

Exercise 5 (Mon 11.10, 10-12, C222)

1. **Interval graphs.**

   Is the following an interval graph.



2. **Shortest common superstring and ATSP.**

   Solve the shortest common superstring problem on set $S = \{\texttt{CTTA},\texttt{TGAT},\texttt{TACT},\texttt{GATG}\}$ by reducing the problem to asymmetric traveling salesman problem through the prefix graph and dummy vertex as described at the lecture.

3. **Shortest common superstring and minimum weight cycle cover.**

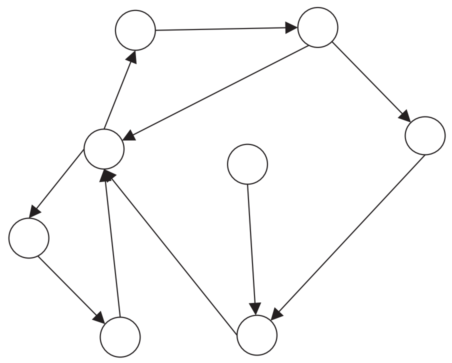
   Simulate the 4-approximation algorithm for shortest common superstring problem on the same set S as above. Visualize also the minimum weight perfect matching corresponding to the minimum weight cycle cover. What is the real approximation factor achieved on this instance?

4. **Preprocessing for gene rearrangement study.**

   Consider you have the genome sequences of two species A and B and you would like to study their rearrangement distance. Each gene in A may have several putative homologs with different local alignment score in B, and vice versa. How would you find a one-to-one mapping between all genes in A to genes in B so that the sum of the corresponding local alignment scores is maximized? Here we may assume that A has at most as many genes as B (otherwise their role can be switched). *Hint. Reduce to a graph problem and add some dummy nodes/edges.*

5. **Graph editing.**

   Eulerian path in a graph is a path that visits all *edges* exactly ones. Insert and delete minimum number of edges to/from the graph below so that it has an Eulerian path.



Research problem: Approximation algorithm for the shortest approximate superstring problem.

Recall the shortest approximate superstring problem: Find the shortest string that contains an occurrence of each given string in $\mathbf{S}$ within Hamming distance k (see more formal definition in exercise 3). Modify the 4-approximation algorithm to give an approximate solution for the shortest approximate superstring problem. Does the 4-approximation guarantee stay valid? Is the algorithm still polynomial time?

Hint. One way to proceed is to add vertices to the prefix graph that correspond to the Hamming-neighborhood *of each $s \in \boldsymbol{S}$ (all string that are within Hamming distance k from s). The challenge is to build a gadget of dummy nodes/edges and adjust edge weights so that minimum weight cycle cover works as you wish and can still be reduced to a polynomially solvable graph problem (like minimum weight perfect matching).*