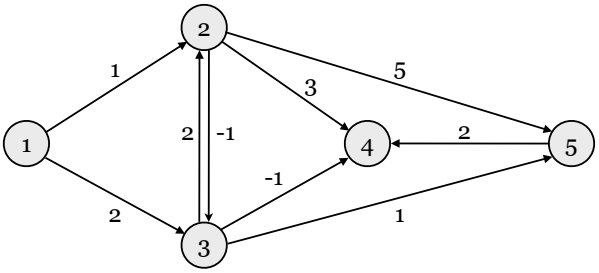
Homework: Introduction

Solutions

1. Consider the graph and the flow vector of the following figure:



(a) Simple paths:

1 arc: (1,2), (1,3).
2 arcs: (1,2,3), (1,2,4), (1,2,5), (1,3,2), (1,3,4), (1,3,5).
3 arcs: (1,2,3,4), (1,2,4,5), (1,2,5,4), (1,2,5,3), ...

Simple forward paths:

1 arc: (1,2), (1,3).
2 arcs: (1,2,4), (1,2,5), (1,3,2), (1,3,5).
3 arcs: (1,2,4,3), (1,2,5,4), (1,3,5,4), (1,3,2,5), (1,3,2,4).
4 arcs: (1,3,2,5,4), (1,2,4,3,5).

(b) Simple cycles:

(1,2,3,1), (1,3,2,1), (1,2,4,3,1), ...

Simple forward cycles:

(2,4,3,2), (2,5,4,3,2), (5,4,3,5).

(c) No, because there is no forward path to node 1, from any other node.

(d) The individual divergence for each node are:

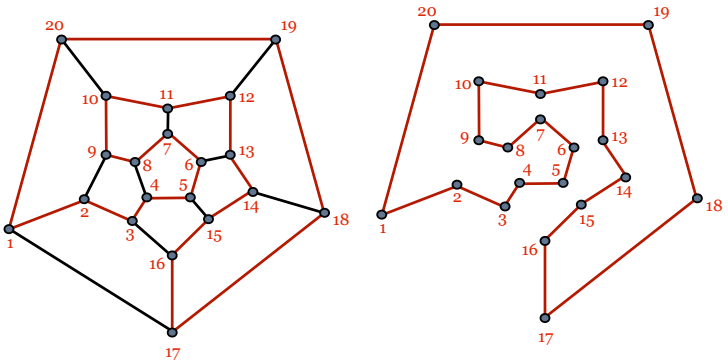
$$
\begin{aligned}
y_1 &= 3 \\
y_2 &= (5+3)-(1+2+1) = 4 \\
y_3 &= (2+1+1)-(2+1) = 1 \\
y_4 &= 1-(3+2) = -4 \\
y_5 &= 2-(5+1) = -4
\end{aligned}
$$

$$
\sum_{i=1}^{5} y_i = 0.
$$

(e) (1,2,4,3,5).

2. **(a)** A Hamiltonian cycle is a simple forward cycle which contains *all the nodes* of the graph.

(b) A Hamitonian cycle on the graph below is:

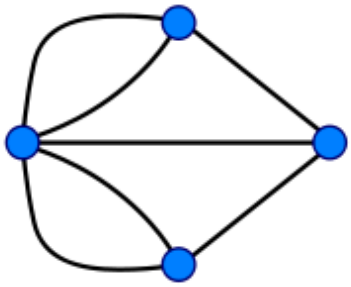


(c) Acyclic is considered a graph that contains no simple cycles.

3. **(a)** Euler cycle (or Eulerian cycle or Euler tour) is a cycle that uses every arc of the graph exactly once.

(b) Euler path (or Euler walk) is a path that uses every arc of the graph exactly once.

(c) The problem of the bridges of Königsberg requires in finding an Euler path. The city of Königsberg can be represented by the following undirected graph,



where the nodes are lands and arcs are bridges.

Euler's argument shows that a necessary and sufficient condition for the walk of the desired form is that the graph be connected and have exactly zero or two nodes of odd degree. The Knigsberg Bridges graph does not have a Eulerian path; therefore, a solution does not exist.

An alternative form of the problem asks for a path that traverses all bridges and also has the same starting and ending point (Euler circuit/tour/cycle). Such a cycle exists if, and only if, the graph is connected, and there are no nodes of odd degree at all.

4. The new formulation now becomes:

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{j|(i,j)\in\mathcal{A}} x_{ij} - \left(\sum_{j|(j,i)\in\mathcal{A}} x_{ji} + x_{Bi} \right) = \underline{s}_i, \qquad \forall i \in \mathcal{N} \\
& \sum_{j|(B,j)\in\mathcal{A}} x_{Bj} = -\sum_{i\in\mathcal{N}} \underline{s}_i, \qquad \text{(for the total divergence to be zero)}, \\
& b_{ij} \le x_{ij} \le c_{ij}, \qquad \forall (i,j) \in \mathcal{A}, \\
& 0 \le x_{Bi} \le \bar{s}_i - \underline{s}_i, \qquad \forall (B,i) \in \mathcal{A},
\end{aligned}$$

5. Construct the bipartite graph $\mathcal{G} = (\mathcal{N}_M \cup \mathcal{N}_W, \mathcal{A})$ in which nodes in $\mathcal{N}_M$ correspond to men, nodes in $\mathcal{N}_M$ correspond to women, and an arc (i, j) exists in $\mathcal{A}$ if, and only if, the man corresponding to node i is compatible with the woman corresponding to node j. Solve a *maximum cardinality matching problem* (i.e., maximize the number of pairs of nodes matched) in the graph $\mathcal{G}$. The arcs in the matching correspond to the couples in the actual matching. The formulation is therefore,

$$\begin{aligned}
\text{maximize} \quad & \sum_{(i,j)\in\mathcal{A}} (a_{ij}a_{ji})x_{ij} \\
\text{subject to} \quad & \sum_{j|(i,j)\in\mathcal{A}} x_{ij} = 1, \qquad \forall i \in \mathcal{N}, \\
& \sum_{i|(i,j)\in\mathcal{A}} x_{ij} = 1, \qquad \forall j \in \mathcal{N}, \\
& 0 \le x_{ij} \le 1, \qquad \forall (i,j) \in \mathcal{A},
\end{aligned}$$

6. **(a)** After the 1st pass, the element at the n-th position will be the longest element. After the 2nd pass, the element at the $(n-1)$-th position will be the next longest element. In general, after k passes, the elements from the $(n-k+1)$-th position to the nth position will form a sorted list of the k longest elements. Hence, after n passes, the algorithm will terminate. Since each pass requires $O(n)$ comparisons, the algorithm will run in $O(n^2)$ time.

(b) The algorithm may be proved to be $\Omega(n^2)$ by simulating it on the following pathological problem instance: $\{2, 3, 4, ..., n, 1\}$. In each pass, the element 1 will move to left by one position. After $(n-1)$ passes, this element moves to the first place, and a sorted list is obtained. The total number of comparisons in all passes are:

$$(n-1) + (n-2) + ... + 1 = \Omega(n^2).$$

Consequently, the bubble sort algorithm takes $\Theta(n^2)$ time.