Assignment 3
Algorithms for Big Data

Lavinia Pulcinella
Student ID: i6233926

March 2020

1 Graded exercise 3.2

Consider the problem of computing an ordering of a set V of n vertices that satisfies the maximum number of given m pairwise-order constraints $C \in V \times V$, where a pairwise constraint $(a,b) \in C$ is satisfied by an ordering $\Psi : V \to \{1,2,...,n\}$ (a permutation of V), if $\Psi(a) < \Psi(b)$. That is, a pairwise-constraint (a, b) expresses the property "a is to the left of b".
Show that the following algorithm is a $\frac{1}{2}$ -approximation algorithm and that it can be implemented in linear-time, i.e., in time $O(n+m)$: consider an arbitrary ordering, and its reverse; output the better of the two orderings.

In order to solve the problem we should take into account the number of constraints that are satisfied and those which aren't. Thus, let m_1 be the number of satisfied constraints and in contrast m_2 is the number of those that are not.

We now consider and arbitrary ordering Ψ let a be on the left of b such that $\{a,b\}$ leads to the constraint $\Psi(a) < \Psi(b)$ to be satisfied and thus it can be counted in the m_1 number of satisfied constraint.
On the contrary, considering the reversed ordering, let Ψ' be the reversed ordering of Ψ. In this case, with b being on the right of a we have an "opposite" constraint. Hence, all constraints that don't satisfy Ψ can satisfy Ψ'.
In Ψ' the total number of constraints can be denoted m' where $m'_1 = m_2$ and $m'_2 = m_1$.

Thus, in the optimum case we would have $m_1 > m_2$ so $m_1 > \frac{m}{2}$. Similarly, $m'_1 > m_2 = m'_1$ so $m'_1 > \frac{m}{2}$. It follows that we need at least $\frac{m}{2}$ constraints and hence the algorithm is a $\frac{1}{2}$-approximation algorithm.

If the pairwise constraint is satisfied i.e. if $\Psi(a) < \Psi(b)$ then m_1 is updated by one. Otherwise, we update m_2. Moreover, in order to select the best output, if $m_1 = \frac{m}{2}$ return the ordering algorithm otherwise its reverse.

Looping through the n vertices requires $O(n)$ time while the counting of the constraints requires $O(m)$ time. Thus, for a total $O(n+m)$ running time.