Exercise set V, Algorithms and Networks 2017-2018

Please hand in your solutions either by email to H.L.Bodlaender@uu.nl or on paper during the course or in the mailbox of Hans Bodlaender in the coffeeroom of the 5th floor of the Buys Ballot Gebouw. Note that you may work alone or work in pairs, see the website for further details. The deadline for this exercise set is Monday, January 8th 2018, 23:59.

In case you hand in your work by email:

- Have [AN5] in the header of the email.
- Attach ONE pdf-file to your email. (If you use Word, note that Word can output the text as pdf-file.)
- Make sure that your name(s) and student number(s) is/are in the pdf-file.

You can write either in English or in Dutch. Your work should be well phrased, readable, understandable, and of course, correct. It is not your teachers job to figure out what you might mean with what you have written. As such, both messy work and work that is difficult to understand can be graded with a 0.

1. Counting numbers (1 points)

How many integers in the range $[1, 2, \ldots, 10000]$ are either divisible by 5, 11 or 19?

2. Inclusion/Exclusion for counting perfect matchings (2 points)

A *perfect matching* in a graph $G = (V, E)$ is a subset of the edges $M \subseteq E$ such that every vertex $v \in V$ is incident to exactly one edge in M. Even though it is possible to find a perfect matching a graph G if it exists in polynomial time, the problem of counting the number of perfect matchings in a graph G is hard (it is $\#P$-complete).

In this exercise, we will construct an algorithm for this problem that uses $\mathcal{O}^*(2^n)$ time and polynomial space. We note that faster algorithms exist.

(i) (0.5 points) Show that the number of perfect matchings in a graph G equals the number of edge subsets $M \subseteq E$ of size $n/2$ with the property that every vertex $v \in V$ is incident to *at least one* edge in M.

(ii) (0.5 points) Show that for a given vertex subset $U \subseteq V$, the number of edge subsets $F \subseteq E$ of size $n/2$ in $G[U]$ can be computed in polynomial time (there exist a direct formula for this number). Note that the edge subsets counted in this subquestion are not perfect matchings.

(iii) (1 point) Show how to use (i) and (ii) and the principle of inclusion/exclusion to construct $\mathcal{O}^*(2^n)$ time and polynomial space algorithm for counting perfect matchings.

Algorithm 1 mis(G)

Input: *A graph $G = (V, E)$ with maximum degree three.*
Output: *The maximum size of an independent set in G.*
if G is empty **then**
  Return 0
else if there exists a vertex $v \in V$ with $d(v) = 0$ **then**
  Return $1 +$ mis$(G - \{v\})$;
else if there exists a vertex $w \in V$ with $d(w) = 1$ **then**
  Let G' be obtained by removing w and its neighbour.
  Return $1 +$ mis(G')
else if all vertices in G have degree two **then**
  Return the size of a maximum independent set of G (computed in polynomial time)
end if
Choose a vertex $x \in V$ of maximum degree.
Let G^1 be obtained from G by removing x.
Let G^2 be obtained from G by removing x and all neighbours of x.
Return max{mis(G^1),mis(G^2) + 1}

3. Independent set and non-standard measures (5 points)

Consider the following algorithm for Independent Set on graphs of maximum degree three.

(i) (1 points) Explain how we can compute in polynomial time the maximum size of an independent set of a graph where all vertices have degree two.

(ii) (0.5 points) Explain why the given algorithm (mis) is correct.

In order to analyze the running time of the algorithm better, we will use a non-standard measure. Suppose we give every vertex of degree 0 or 1 weight 0, every vertex of degree two weight $\frac{1}{2}$, and every vertex of degree three weight 1. The *measure* of a graph G then is the total weight of all vertices in G. We denote this by $m(G)$.

(iii)(0.5 points) Consider the graph G^1 from the algorithm, i.e., the graph obtained by removing the vertex x in the branching step. Argue that $m(G^1) \leq m(G) - \frac{5}{2}$. (Hint: look what happens to the weights of the neighbors of v.)

(iv) (0.5 points) Consider the graph G^2 from the algorithm, i.e., the graph obtained by removing the vertex x and its neighbors in the branching step. Argue that $m(G^2) \leq m(G) - \frac{5}{2}$.

(v) (0.5 points) Prove that the algorithm uses $O^*(c^n)$ time with c the positive solution of the equation

$$c^{\frac{5}{2}} - 2 = 0$$

(vi) (0.5 points) Give an upper bound on the running time of the algorithm that follows from (v) expressed as $O^*(c^n)$ where x is an explicit decimal number. Briefly explain how you obtained your result.

(vii) (1 point) Improve upon the claimed result of step (iv) by a better case analysis, and looking at vertices at distance two from G. (It is easy to make a mistake here; note that the neighbors of v can have edges to each other ...)

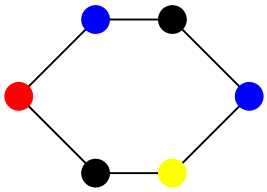
(viii) (0.5 points) Use your improved result from (vii) and improve upon the upper bound you obtained in (vi).

4. Graph Isomorphism (2 points)

(i) Give two connected, undirected graphs with 4 vertices that are not isomorphic. (0.5 point)

(ii) Suppose we have two colored cycles, both with n vertices. I.e., we have a set of colors $\mathcal{C}$, and each vertex has a color from $\mathcal{C}$. In an isomorphism for colored cycles, we additionally require that each vertex is mapped to a vertex with the same color.

Show that graph isomorphism for colored cycles can be solved in polynomial time. (1.5 points)



Bonus Question: Yet another measure for independent set (1+1 points)

Reconsider the algorithm $mis(G)$ for maximum independent set on graphs of maximum degree 3 from the exercise above. Prove an even better bound on the running time of the algorithm by using a different measure? That is, by not using the measure $m(G)$ specified in the exercise but yet another measure. The best upper bound proved by any student receives the second bonus point.

Research Question (possibly very difficult): Exponential-time algorithm for capacitated vertex cover

A *vertex cover* in a graph $G = (V, E)$ is a subset of the vertices $C \subseteq V$ such that every $e \in E$ is incident to a vertex in C.

Consider a graph $G = (V, E)$ and for every vertex $v \in V$ a capacity $c_v \in \mathbb{N}$. In the Capacitated Vertex Cover problem, the capacities allow a vertex v to cover at most c_v edges (edges to which v must be incident). That is, a *capacitated vertex cover* $C \subseteq V$ in G is a vertex subset such that for every vertex $v \in C$ there exists a subset of the edges incident to v that we call E_v, with $|E_v| \leq c_v$, such that $\cup_{v \in C} E_v = E$.

Open research problem: Find an algorithm for capacitated vertex cover running in time $\mathcal{O}^*((2 - \epsilon)^n)$.