Technische Universität
Kaiserslautern

Department of Computer Science
AG Algorithms and Complexity

Issue Date: 08.05.2013
Version: 2013-05-23 18:18

4th Exercise sheet for Advanced Algorithmics, SS 13

Hand In: *Until Wednesday, 15.05.2013, 12:00am, Exercise sessions, hand-in box in stairwell 48-6 or email.*

Problem 6

Solve the following inhomogenous linear recurrence equations using generating functions:

a)
$$\begin{aligned} a_0 &= 4, \\ a_{i+1} &= 2a_i + 3^i \quad , i \geq 0 \, . \end{aligned}$$

b)
$$\begin{aligned} b_0 &= 2, \\ b_1 &= 2, \\ b_i &= 6 \cdot b_{i-1} - 8 \cdot b_{i-2} + 13 \cdot i \quad , i \geq 2 \, . \end{aligned}$$

Problem 7

Consider the following problem:

Input: A graph $G = (V, E)$ and $k \in \mathbb{N}$.

Question: Can we transform G, by deleting or adding at most k edges, into a graph that consists of a disjoint union of disconnected cliques?

Turn page!

a) Consider the following algorithm for the problem:

Given $G = (V, E)$ and $k \in \mathbb{N}$, do the following:

1) If G is already a union of disjoint cliques, we are done: report "yes" and return.
2) Otherwise, if $k \leq 0$ we can not find a solution in this branch. Report "no" and return.
3) Otherwise, identify $u, v, w \in V$ with $\{u, v\} \in E$ and $\{u, w\} \in E$, but $\{v, w\} \notin E$. Call the algorithm on three instances $((V, E'), k')$ defined by

   (B1) $E' = E \setminus \{\{u, v\}\}$ and $k' = k - 1$,

   (B2) $E' = E \setminus \{\{u, w\}\}$ and $k' = k - 1$ and

   (B3) $E' = E \cup \{\{v, w\}\}$ and $k' = k - 1$,

   respectively. Report "yes" if at least one of the recursive calls reports such, and "no" otherwise.

Show that this algorithm solves the given problem and give a non-trivial upper bound on its runtime as $\mathcal{O}$-class.

b) In order to improve the algorithm given in a), we can distinguish three cases that can occur for any chosen "*conflict triple*" (u, v, w) as specified in step 3:

(C1) Vertices v and w do not share a common neighbour, that is

$$\forall x \in V \setminus \{u\} : \{v, x\} \notin E \vee \{w, x\} \notin E \,.$$

(C2) Vertices v and w have a common neighbour $x \neq u$ and $\{u, x\} \in E$.

(C3) Vertices v and w have a common neighbour $x \neq u$ and $\{u, x\} \notin E$.

It is possible to show that we can restrict ourselves to the following branching.

- In case (C1), we only have to take branches (B1) and (B2).
- In case (C2), we have to execute (B1) and refine the other two branches further so that each has one subbranch that deletes one, and another that deletes two additional edges.
- In case (C3), we have to execute (B1). Branch (B2) can be refined into one branch that deletes an additional edge, and one that adds and deletes one edge, respectively. Branch (B3) can be refined into one branch that deletes two additional edges, and one that adds one edge, respectively.

Give a non-trivial worst-case bound on the size of the search tree as explored by this improved algorithm!