CPSC 413 June 4, 2019

University of Calgary

Homework Assignment # 3

Due: June 18, 2019, **11:59am (noon)**

LATE ASSIGNMENTS WILL NOT BE ACCEPTED

Please fill out the cover page at the end of this assignment and attach it to your solutions.

Collaboration Rules:

In order to solve the questions, you are allowed to collaborate with students from the same course, but not with anybody else. If you do collaborate with other students, you have to list them in the appropriate fields on the cover page. You have to write up the solutions *on your own* in *your own words*. You can meet with your collaborators in order to generate ideas, but you are not allowed to write up the solutions during these meetings or to take any notes, pictures, etc. away from those meetings.

You are allowed to use literature, as long as you cite that literature in a scientific way (including page numbers, URLs, etc.). In any case, your solutions must be self-contained. For example, if you use a theorem or lemma that was not covered in the lecture and that is not "well-known", then you have to provide a proof of that lemma or theorem.

If you are in doubt whether a certain form of aid is allowed, ask your instructor!

Academic misconduct (cheating, plagiarism, or any other form) is a very serious offense that will be dealt with rigorously in all cases. A single offense may lead to disciplinary probation or suspension or expulsion. The Faculty of Science follows a zero tolerance policy regarding dishonesty. Please read the sections of the University Calendar under the heading "Student Misconduct".

Question 1

The problem $LONGPATH_{search}$ is to find the longest simple path in a given undirected graph.

(a) Define the decision and the optimization variant of the $LONGPATH_{search}$ problem.

$LONGPATH_{search}$
Input: An undirected graph G=(V, E).
Output: The longest simple path in graph G.

$LONGPATH_{opt}$
Input: An undirected graph G=(V, E).
Output: The maximum integer k, that corresponds to the length of the longest possible simple path in G.

$LONGPATH_{dec}$
Input: An undirected graph G=(V, E) and a variable k.
Output: "Yes", if there exists a simple path in G that is at least k edges long, and "No" otherwise.

(b) Show that all three variants of this problem are polynomial-time equivalent.

Show that LONGPATHdec =P LONGPATHopt =P LONGPATHsearch

Thus we have to prove

$LONGPATH_{dec} \leq P\ LONGPATH_{opt} \leq P\ LONGPATH_{search} \leq P\ LONGPATH_{opt} \leq P\ LONGPATH_{dec}$

"$LONGPATH_{dec} \leq P\ LONGPATH_{opt}$":

Let B_{opt} be a black box for $LONGPATH_{opt}$.

The following algorithm determines whether G has a path of at least size k.

1. Ask B_{opt} about G and let m be the answer
2. If $m \geq k$, accept, otherwise reject.

Correctness is obvious.

"$LONGPATH_{opt} \leq P\ LONGPATH_{search}$":

Let B_{search} be a black box for $LONGPATH_{search}$.

1. Ask B_{search} about G and let S be the answer (the longest simple path).
2. Output |S|

Correctness is obvious

"$\underline{\text{LONGPATH}_{search} \leq_P \text{LONGPATH}_{opt}}$":

Let B_{opt} be a black box for LONGPATH_{opt} .

The following algorithm determines a simple path with the longest value in graph G.

Function LONGPATH_{search}(G = (V; E))

Determine the size k of a maximal independent set of G by asking B_{opt}.

if |E| = k then

return E

forall edges "e" element of E do

Let U = E \{e}

Ask B_{opt} to determine the size k' of U

if k' = k then return LONGPATH_{search}(U)

Running time: T(0) = O(1), T(n) = T(n 1) + O(n).

T(n) = O(n^2)

Correctness:

This algorithm works because in every iteration an edge is going to be eliminated if it is not part of the optimal solution for any test case, if an element is deleted, then it means that the algorithm still needs to run in order to reduce any unnecessary edges to our solution.

"$\underline{\text{LONGPATH}_{opt} \leq_P \text{LONGPATH}_{dec}}$":

Let Bdec be a black box for LONGPATH_{dec}

Function LONGPATHopt(G(V,E), k):

for k= n .... 0 do

Ask B_{dec} whether G has an simple path of size k

If the answer is yes, then return (k)

Correctness and polynomial running time are obvious.

THUS we have proven LONGPATHdec =P LONGPATHopt =P LONGPATHsearch

Question 2

The problem EQUAL_SPLIT is to decide for a given multi-set S of positive integers if there is

$M \subseteq S$, such that $$\sum_{x \in M} x = \sum_{x \in S-M} x.$$

The problem EXACT is to decide for a given multi-set S of positive integers and an integer k, if there is $M \subseteq S$, such that

$$\sum_{x \in M} x = k.$$

Prove that EQUAL_SPLIT and EXACT are polynomial-time equivalents.

Firstly, we reduce EXACT to EQUAL_SPLIT:

Let (A,k) be an arbitrary instance of EXACT. We transform this into an instance A' of the EQUAL_SPLIT problem in this way:

- Let m = (the sum of the elements in A).
- Let A' be the set A plus two more elements:
  - a_{n+1} = $(2m+k)$, and
  - a_{n+2} = $(3m-k)$.

Prove of correctness:

A' can be constructed easily in polynomial time.

We claim that A' is a true instance of EQUAL_SPLIT if and only if (A,k) is a true instance of EXACT. Assume (A,k) is a true instance of EXACT. Then there is a subset S of A such that S has size k. Then $((A\text{-}S)$ union $\{a_{n+1}\}$, S union $\{a_{n+2}\})$ is a Subset of A'.

Now assume that A' is a true instance of EQUAL_SPLIT. Then there exists an equal-weight partition (S_1, S_2) of A'. Notice that the sum of all the elements in A' is 6m. So both a_{n+1} and a_{n+2} cannot appear in the same subset. Without loss of generality, we say that a_{n+1} is in S_1 and a_{n+2} is in S_2. Then by construction S_2 - $\{a_{n+2}\}$ has sum k (since S_2 has weight $3m$). It follows that there is a subset of weight k in A.

Therefore, The given reduction reduces EXACT to EQUAL_SPLIT.

Secondly, we reduce EQUAL_SPLIT to EXACT:

Let B_{exact} be a black box for EXACT

Let m be the sum of our input set S

If m is divisible by 2 then

    Let k=m/2

    Run $B_{exact\ (S,k)}$

    If Bexact (S,k) returns true

        return “yes”

    else

        return “no”

else

    return “no”

Correctness:

The algorithm will always return the correct answer since we will first divide the Set of S into two subsets of equal value (if a division by two is possible) and finally, our algorithm will run the Bexact black box and determine if from the current set and the value that k= equal the total sum of the entire set, if we can find a set that equals k =(which is the sum of the rest of the elements if there is any).

Question 3

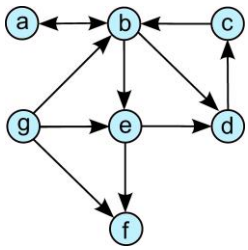
The problem *fully connected subgraph*, short (FCS) is to decide for an undirected graph $tt = (V, E)$ and an integer k if there is a set $U \subseteq V$ of at least k vertices, such that any two distinct vertices in U are adjacent in tt.

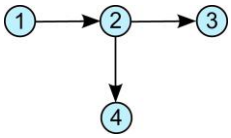
The problem *Subgraph Matching*, short SM, is to decide for two directed graphs $tt_1 = (V_1, E_1)$ and $tt_2 = (V_2, E_2)$ whether tt_2 is isomorphic to a subgraph of tt_1. tt_2 is isomorphic to a subgraph of tt_1, if there is a mapping $f: V_2 \rightarrow V_1$, s.t.

$$\forall (u, v) \in E2 : (f(u), f(v)) \in E1.$$

Informally, this means that in tt_1 you can remove some vertices and edges such that the remaining graph looks exactly like tt_2 (ignoring the vertex labels).

For example, tt_2 below is isomorphic to a subgraph of tt_1 below. The corresponding mapping is $f(1) = g, f(2) = e, f(3) = d$, and $f(4) = f$.





Prove that FCS <=P SM

Firstly,
Let Bsm be a black box for SM
(We will use this when running FCS)

Function FCS (G(V,E), k)
Let G'(V', E') be a temporary graph

for every edge u of E do
Transform every undirected edge u connecting G to a bidirectional directed edge in G'
(Add the edge to E' and add the connected vertices as well)

Let G''(V'', E'') be a secondary graph (we will store the subgraph here)
for every i = 0.....k
We will add adjacent vertices from E' to E'' until we create a subgraph G'' (from G') with size k

If there are no more edges to add before k
return "no"

Run Bsm(G', G'') //run SMblackbox with the original graph and the obtained subgraph as inputs

If Bsm black box return true
return "yes"
else
return "no"

Correctness:

Coming from the FCS ALGORITHM we obtain the Graph G and the required k to test if there is a fully connected graph of that length. Firstly we convert our original Graph to a directed graph by replacing every undirected edge into the bidirected edge and store it into a new temp Graph. Then we create a new graph if there is any fully connected graph with at least size k from the newly created bidirected graph, OTHERWISE, return false.

(If the graph we end up with is at least k edges long) We would then use the Bsm black Box to determine if the G' bidirected graph and The subgraph we obtain in the end returns true if both graph G" is isomorphic to a subgraph of G; given us the reduction from SM to FCS.

Cover Page for CPSC 413
Homework Assignment # 3

Name: **Juan Luis de Reiset Jimenez-Carbo**

Course Section: CPSC 413 TUT-02

Collaborators:

Question 1: Daniel Sohn, Steve Khanna, Coskun Sahin

Question 2: Daniel Sohn, Steve Khanna, Coskun Sahin

Question 3: Daniel Sohn, Steve Khanna, Coskun Sahin

Question 4: Daniel Sohn, Steve Khanna, Coskun Sahin

Other Sources:

Question 1:

Question 2: https://www.cs.mcgill.ca/~jmerce1/a4ans.html?fbclid=IwAR0nSoYOADejeDnBXwX5q_cU43-r3cShlzDxxyBOO0JaVsz62zDjho-qT_w

Question 3:

Question 4:

Declaration:

I have written this assignment myself. I have not copied or used the notes of any other student.

Date/Signature:

Juan Luis de Reiset Jimenez-Carbo
UCID:30050167

6/17/2019