Assignment (Bounded Treewidth Algorithms)

Due: April 12, 11:59pm

Requirements

In this assignment you will learn how to use and understand the t-parse data structure for graphs of bounded pathwidth/treewidth. We will do some basic programming (three problems) regarding the t-parse representation of graphs.

We use the following t-parse input format for graphs of bounded treewidth. The first token will be a positive integer $t \leq 9$ denoting the pathwidth/treewidth of the graph (i.e., it indicates a t-parse will follow). The interpretation for the remaining tokens are given in the next table.

token	**semantic meaning**
i	a vertex operator i, represented as an integer, where $0 \leq i \leq t \leq 9$
ij	an edge operator (i, j) where $t \geq i > j \geq 0$ (note: no space between i and j)
(	a begin marker for a pathwidth t-parse (can be nested for treewidth t-parses)
)	an end marker for a pathwidth/treewidth t-parse

Thus, if you read/encounter the two tokens ')' and '(' in sequence then this denotes a circle plus $\oplus$ operator. Note that it is guaranteed that each right token '(' will have a matched left token ')'. Each '(' except first token must be preceded by a parenthesis and ')' may be succeeded "up the parse tree" by pathwidth operators. We assume boundary vertices $0, 1, \ldots, t$ precede the first token of a pathwidth t-parse (and these empty boundaried graph axioms are not given in the input format). Assume *left-associativity* for all operators and at least one space between each of them, including the parentheses.

For each part of the assignment the input will consist of several test cases, each on its own line. The input is terminated by a $t = 0$ case, which is also processed.

Submission and Marks

All solutions should be submitted to the automarke.
All input should be taken from the keyboard (stdin) and output to the console (stdout). There will be two test cases for each of our problems.

The first data set will only contain pathwidth t-parses and are worth 3 marks. The slightly harder treewidth t-parse input cases will be worth 2 marks each. This assignment is worth 15% of your final grade, where each of the three problems are worth 5 marks. Partial marks may be awarded later (after due date) if determined to be close to correct.

Problem 1: Parse and Extract Graph

Here the first line is an integer n representing the order of the graph. The next n lines consist of a (sorted integer) adjacency lists, for vertices indexed 0 to $n-1$, of neighbor indices. To have unique graph, the vertex indices are assigned labels to vertices in the same order as the new vertex operators appear in the t-parse from left-to-right.

The following show typical input instances and expected output. Note that you can assume that each input graph fits on one line.

Sample input
2 (10 21 1 10 21 0 10 20 2 20 21)
2 ((10 21 1 10 21 1 21 10) (21 2 21 20) (10))
4 ((10 20 30 41 4 41 31) ((40 21 32 42) (10 0 43 10 20)))
0 ((0) (0 0))

Sample output
6
1 3
0 2
1 3 4
0 2 4 5
2 3 5
3 4
6
1 2 3 4
0 2
0 1 3 4
0 2
0 2 5
4
7
1 2 3 5
0 2 3 4 5 6
0 1 3 5
0 1 2 5
1
0 1 2 3
1
4

Problem 2: Sorted Degree Sequence

You need to read in each graph and output its **degree sequence** (space separated) in decreasing sorted order. Note that we are interested in the degrees of the vertices of a simple graph (multi-edges should be ignored).

Note you can either use your solution for Problem 1 to easily get the degree sequence or write an algorithm that directly extracts the degree sequence from a t-parse. The second approach might help you with Problem 3.

The following show typical input instances. Note that you may either assume each input graph fits on one line.

Sample input
2 (10 21 1 10 21 0 10 20 2 20 21)
2 ((10 21 1 10 21 1 21 10) (21 2 21 20))
4 ((10 20 30 41 4 41 31) ((40 21 32 42) (10 0 43 10 20)))
3 (32 31 30 20 21 10 0 10 20 21 1 10 21 1 10 21 1 10 0 10 30 20)
2 (21 2 21 2 21 20 1 10 1 10 1 21 1 10 2 21 1 10 21 2 21 1 10 2 20 0 20 10)
3 ((30 32 21 10 1 10 21 2 21 32 1 21 10 31 1 21 10 31) (20 21 32 2 21 32) 0 21 10 30 20)
4 (10 20 30 40 43 21 32 2 20 21 42 0 30 40 10 2 21 3 30 31 4 41 43 42 40)
0 (0)

Sample output
4 3 3 2 2 2
4 4 3 2 2 1
6 4 4 4 4 1 1
7 5 4 4 3 3 2 2 2
7 3 3 3 2 2 2 2 2 1 1 1 1 1 1
7 6 5 5 3 3 3 3 3 2
7 5 5 4 4 4 3 3 3 2
0 0

Problem 3: Weighted Independent Sets

In this part of the assignment you will learn how to exploit the structure of bounded pathwidth/treewidth for a known hard graph problem:

Problem: Degree-Weighted Independent Set (DWIS)

Input: (undirected) Graph $G = (V, E)$

Question: Let the weight of each vertex be its degree (number of vertex neighbors).
What is the largest sum of vertex weights over all independent sets of vertices?
That is, $\max_{V' \subseteq V} \sum_{v \in V'} \deg(v)$ where for all $\{v_1, v_2\} \subseteq V'$ we have $(v_1, v_2) \notin E$.

Your main requirements are to do the following.

1. For input graph of bounded pathwidth (in t-parse representation described above), determine the DWIS.
2. Extend your linear-time program for item 1 for input of bounded treewidth.

The following show typical input instances. Again you can assume each input t-parse graph fits on one line.

Sample input	**output**
2 (10 21 1 10 21 0 10 20 2 20 21)	7
2 ((10 21 1 10 21 1 21 10) (21 2 21 20))	7
4 ((10 20 30 41 4 41 31) ((40 21 32 42) (10 0 43 10 20)))	6
3 (32 31 30 20 21 10 0 10 20 21 1 10 21 1 10 21 1 10 0 10 30 20)	11
2 (21 2 21 2 21 20 1 10 1 10 1 21 1 10 2 21 1 10 21 2 21 1 10 2 20 0 20 10)	16
3 ((30 32 21 10 1 10 21 2 21 32 1 21 10 31 1 21 10 31) (20 21 32 2 21 32) 0 21 10 30 20)	14
4 (10 20 30 40 43 21 32 2 20 21 42 0 30 40 10 2 21 3 30 31 4 41 43 42 40)	13
0 (0 0)	0