Exam EDAF05

26 May 2014, 8–13, Vic2

Thore Husfeldt

Instructions

What to bring. You can bring any written aid you want. This includes the course book and a dictionary. In fact, these two things are the only aids that make sense, so I recommend you bring them and only them. But if you want to bring other books, notes, print-out of code, old exams, or today's newspaper you can do so. (It won't help.)

You can't bring electronic aids (such as a laptop) or communication devices (such as a mobile phone). If you really want, you can bring an old-fashioned pocket calculator (not one that solves recurrence relations), but I can't see how that would be of any use to you.

Filling out the exam Some questions are multiple choice. Mark the box or boxes with a cross or a check-mark. If you change your mind, completely black out the box and write your answer's letter(s) in the left margin. In case it's unclear what you mean, I will choose the least favourable interpretation.

In those questions where you have to write or draw something, I will be extremely unco-operative in interpreting your handwriting. So *write clearly*. Use English or Swedish. If there is a way to misunderstand what you mean, I will use it.

Scoring For the free form choice questions, you get between 0 and the maximum number of points for that question. Short, correct answers are preferred.

Each multiple choice question has exactly *one* correct answer. To get the maximum score for that question, you must check that answer, and only that. However, to reflect partial knowledge, you *may* check several boxes, in case you're not quite sure (this lowers your score, of course – the more boxes you check, the fewer points you score). If you check *no* boxes or *all* boxes, your score for that question is 0. If the correct answer is not among the boxes you checked, your score is negative, so it's better to *not* answer a question where you're on very thin ice. The worst thing you can do is to check all boxes except the correct one, which gives you a large negative score.

Want an example? Assume a question worth maximum 2 points has $k = 4$ possible answers (one of them correct).

- If you select only the correct answer, you receive 2 points.
- If you select 2 answers, one of which is correct, you receive 1 point.
- If you select 3 answers, one of which is correct, you receive 0.41 points.
- if you select no answer or all answers, you receive 0 point.
- If you select only one answer, and it is wrong, you receive -0.67 points.
- If you select 2 answers that are both wrong, you receive -1 point.
- If you select 3 answers that are all wrong, you receive -1.25 points.

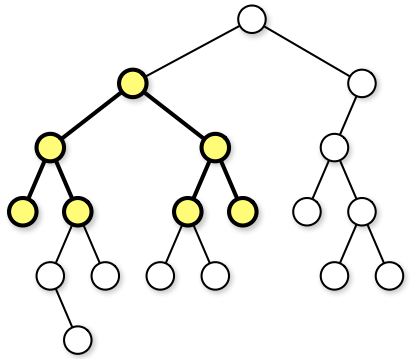
As a special case, for a yes/no question, you receive 1, 0, or -1 points, depending on whether your answer is correct, empty, or wrong.

If you want to know the precise formula: if the question has k choices, and you checked a boxes, your score is $\log(k/a)$, provided you checked the correct answer, and $-a \log(k/a)/(k-a)$ if you only checked wrong answers. Moreover, I have weighted the questions by relevance (not necessarily difficulty), and indicated the maximum points with each question.

You really care why this scoring system makes sense? Then read [Gudmund Skovbjerg Frandsen, Michael I. Schwartzbach: A singular choice for multiple choice. SIGCSE Bulletin 38(4): 34–38 (2006)]. For example, random guessing will give you exactly 0 points, at least in expectation.

Subtree

In this exercise, we consider a rooted binary tree G.



In particular, every vertex has 0, 1, or 2 children, there are no cycles, and G is connected. There is a single root.

Our task is to find a largest perfect subtree. (Recall that a binary tree is *perfect* if all leaves are at the same depth and every non-leaf has exactly 2 children.) A largest perfect subtree with 7 vertices is shown above.

Input

A description of the tree as a directed graph, oriented towards the leaves.

Output

The depths of a largest perfect subtree. (Let's agree that a single node has depth 1.)

Input:

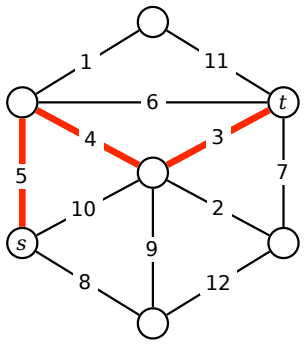
```
1-->2 1-->3 2-->4 2-->5 3-->6 4-->7 4-->8 5-->9 5-->10 6-->11
6-->12 8-- 7-->11 7-->12 8-->13 8-->14 9-->15 9-->16 12-->17
12-->18 13-->19
```

Output:

```
3
```

Clearance

Consider a weighted, undirected graph G, like this one:



The weights are *security levels*. To travel along an edge of security level w you need security clearance $\geq w$. In the above example, you can travel from s to t if you have security clearance 5 or higher. (Walk along the highlighted path.)

The number of vertices is n, the number of edges is m. The weights are n-bit integers.

Input

The edges a weighted graph (endpoints separated by `--`, followed by the weight in parentheses), followed by s and t.

Output

The smallest c so that you can get from s to t using clearance c.

Input:

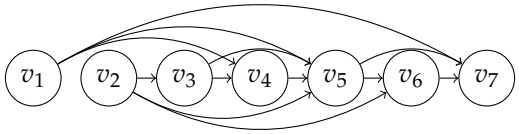
```
1--2 (1)
1--3 (11)
2--3 (6)
2--4 (4)
2--5 (5)
3--4 (3)
3--6 (7)
4--5 (10)
4--6 (2)
4--7 (9)
5--7 (8)
6--7 (12)
5
3
```

Output:

```
5
```

Long

In this exercise, the graph G is directed and that the vertices $v_1, v_2, \ldots, v_n$ are *topologically ordered*. (Recall that this means that if there is a directed edge from from v_i to v_j then $i < j$.) There are m edges.



The task is to find a path of maximum length in G. In the above example, there is a path consisting of 4 edges: v_1, v_4, v_5, v_6, v_7. The longest path in the example consists of 5 edges. (Find it!)

Input

A description of the directed graph G. All edges have weight 1. You can assume it is topologically ordered.

Output

The length of a longest path in G (an integer).

Example:

Input:

```
1 -> 4
1 -> 5
1 -> 7
2 -> 3
2 -> 5
2 -> 6
3 -> 4
3 -> 5
4 -> 5
5 -> 6
5 -> 7
6 -> 7
```

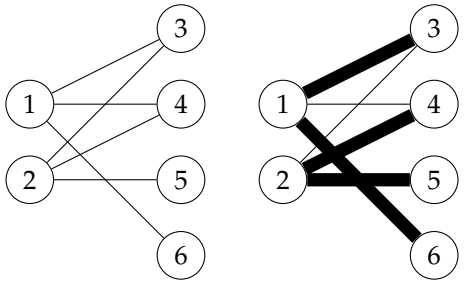
Output:

```
5
```

V

This exercise considers an undirected, unweighted, bipartite graph G with $3n$ vertices. There are n vertices in the left part L and $2n$ vertices in the right part R. No edges connect two vertices in L and no edges connect two certices in R. There are m edges.

The task is to find a *V-matching*. In such a matching, every vertex in L picks exactly 2 vertices in R. No vertex in R is picked twice. (It looks a bit like a bunch of Vs lying on the side, hence the name.) Here's an example for $n = 2$; the input graph is to the left and a V-matching is shown to the right.



Input

The input describes the edges of G.

Output

Yes or no: does G admit a V-matching?

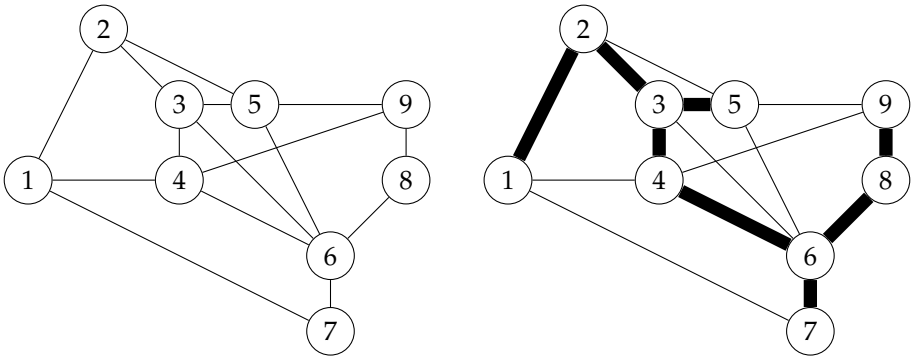
Input:
```
1--3 1--4 1--6 2--3 2--4 2--5
```

Output:
yes

Crossing

In this exercise, G is an unweighted, undirected, connected graph. There are n vertices and m edges.

A *crossing* in a spanning tree is an internal vertex with more than two neighbours. In the example spanning tree to the right, there are two crossing vertices: vertex 3 and vertex 6.



The task is to find a spanning tree with no more than k crossings for given k.

Input

k, followed by a description of the graph. You can assume that the graph is connected.

Output

The edges of a spanning tree with at most k crossings, or `impossible`.

Input:

```
2
1--2 1--4 1--7 2--3 2--5 3--4 3--5
3--6 4--6 4--9 5--6 5--9 6--7 6--8
8--9
```

Output:

```
1--2 2--3 3--4 4--6 6--8 8--9 3--5
6--7
```

Exam Questions

Analysis of algorithms

1.

(a) (*1 pt.*) How many stars are printed?

```
for (int i = N; i > 1; i = i/2) StdOut.print("*");
```

A $O(\log N)$ B $O(N)$ C $O(N \log N)$ D $O(N^2)$

(b) (*1 pt.*) How many stars are printed when I call $f(N)$?

```
static void f(int K)
{  for (int i = 0; i < K; i = i+1) g(i); }

static void g(int K)
{  for (int i = 0; i < K; i = i+1) StdOut.print("*"); }
```

A $O(\log N)$ B $O(N)$ C $O(N \log N)$ D $O(N^2)$

Divide-and-conquer

2. One of the problems in the set can be solved by divide-and-conquer.

(a) (*1 pt.*) Which one?

A Subtree B Clearance C Long D V E Crossing

(b) (*3 pt.*) Describe your solution. Use code, pseudocode, drawings—whatever it takes. Make it clear what the arguments to the recursive call are, what a recursive call returns (it's *type*, at least), and how the returned results are combined.

(c) (*1 pt.*) The total running time is (give the smallest correct estimate)

A $O(\log n)$. B $O(n)$. C $O(n \log n)$. D $O(m \log m)$.

Graph connectivity

3. One of the problems can be solved in polynomial time with a standard graph traversal or connectivity algorithm (BFS, DFS, MST, Dijkstra, connected components, etc.)

(a) (*1 pt.*) Which one?

A Subtree B Clearance C Long D V E Crossing

(b) (*2 pt.*) Describe you algorithm in the box below. Be short and concise. (There is a one-sentence answer. If you do need more space write "see page xxx" and use a separate piece of paper.) State the running time in terms of the original parameters of the problem.

Dynamic programming

4. One of the problems is solved by dynamic programming.

(a) (*1 pt.*) Which one?

A Subtree B Clearance C Long D V E Crossing

(b) (*4 pt.*) Following the book's notation, we let $\mathrm{OPT}(i,j)$ (or $\mathrm{OPT}(i)$? who knows?) denote the value of a partial solution. Then OPT satisfies[1]

A

$$\mathrm{OPT}(i,j) = \begin{cases} 1 + \min\{\mathrm{OPT}(i-1,j), \mathrm{OPT}(i-1,j-1), \mathrm{OPT}(i,j-1)\}\,, & \text{if } c > w, \\ 0\,, & \text{otherwise.} \end{cases}$$

B

$$\mathrm{OPT}(i,j) = \begin{cases} \mathrm{OPT}(i+1,j-1)\,, & \text{if } c > w, \\ 1 + \min\big(\mathrm{OPT}(i+1,j), \mathrm{OPT}(i,j-1), \mathrm{OPT}(i+1,j-1)\big)\,, & \text{otherwise.} \end{cases}$$

C

$$\mathrm{OPT}(i) = 1 + \max_{j>i}\{\,\mathrm{OPT}(j) \colon (i,j) \in E\,\}$$

D

$$\mathrm{OPT}(i,j) = w(i,j) + \max_{j}\{\,\mathrm{OPT}(i-1,j)\,\}$$

E

$$\mathrm{OPT}(i,j) = \min_{j-1 \le k \le j+1}\{\,\mathrm{OPT}(i-1,k) + 1\,\}$$

F

$$\mathrm{OPT}(i,j) = \begin{cases} \mathrm{OPT}(i-1,j) & \text{if } c > w(i,j) \\ \max\{\mathrm{OPT}(i-1,j), 1 + \mathrm{OPT}(i-1,j-1)\} & \text{otherwise} \end{cases}$$

G

$$\mathrm{OPT}(i) = \max\{\mathrm{OPT}(i-1), c + \mathrm{OPT}(i-2)\}$$

(c) (*1 pt.*) State the running time of the resulting algorithm. (Choose the smallest correct estimate.)

A $O(n)$ B $O(m)$ C $O(nm)$ D $O(n^2)$

(d) (*1 pt.*) State the space usage of the resulting algorithm. (Choose the smallest correct estimate.)

A $O(n)$ B $O(m)$ C $O(nm)$ D $O(n^2)$

Network flow

[1] There may be other cases, in particular, boundary conditions such as maybe for $\mathrm{OPT}(1,1)$ or $\mathrm{OPT}(n,0)$ or $i = j$, etc. Don't worry about them. This question is just about the most central part of the recurrence relation, otherwise this exercise becomes too large.

5. One of the problems in the set is easily solved by a reduction to network flow.

(a) (*1 pt.*) Which one?

A Subtree B Clearance C Long D V E Crossing

(b) (*3 pt.*) Draw the resulting network for the example instance. Make sure your draw all nodes, all edges, and all capacities. (I've started by already drawing the source and the sink for you.)

(c) In general, the number of nodes in the resulting graph in terms of the original problem's parameters is: (Choose the smallest correct estimate). (*1 pt.*)

A $O(n)$ B $O(m)$ C $O(nm)$ D $O(n+m)$

Computational complexity

6. One of the problems in the set is NP-hard.[2]

(a) (*1 pt.*) Which problem is it? (Let's call it P_1.)

A Subtree B Clearance C Long D V E Crossing

(b) (*1 pt.*) The easiest way to see that is to take the following NP-hard problem, called P_2,

A Graph colouring B 3-dim. matching C Independent set D Set Cover
E Vertex cover F 3-satisfiability G Travelling salesman H Hamiltonian path

(c) (*1 pt.*) and prove

A $P_1 \leq_P P_2$ B $P_2 \leq_P P_1$

(d) (*5 pt.*) Describe the reduction on a separate piece of paper. Do this both in general and for a small but complete example. In particular, be ridiculously precise about what instance is *given*, and what instance is *constructed* by the reduction, the parameters of the instance you produce (for example number of vertices, edges, sets, colors) in terms of the parameters of the original instance, what the solution of the transformed instance means in terms of the original instance, etc. Please, please, please start your answer with words like this "Given an instance to problem *blabla*, . . .,".

[2] If P = NP then *all* these problems are NP-hard. Thus, in order to avoid unproductive (but hypothetically correct) answers from smart alecks, this section assumes that P $\neq$ NP.