Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

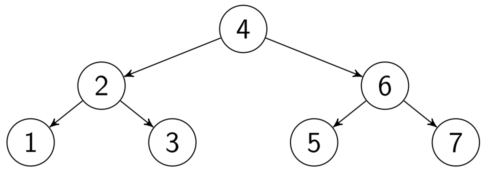
Institut für Theoretische Informatik
Peter Widmayer
Thomas Tschager
Antonis Thomas

20th April 2016

Datenstrukturen & Algorithmen — Exercise Sheet 8 — FS 16

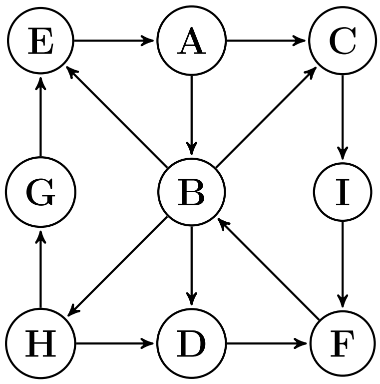
Exercise 8.1 *Optimal search trees and splay trees.*

We are given the keys $1, 2, \dots, 7$ with query frequencies $1, 4, 1, 8, 1, 4, 1$. For example, there is one query for key 3 and 8 queries for key 4. There are no queries for keys in the intervals between the given keys. Compute an optimal search tree and give a sequence of 20 queries (considering the given frequencies), such that there are fewer comparisons in the splay tree (initially as depicted below) than in the optimal search tree.



Exercise 8.2 *Depth-First Search and Breadth-First Search.*

Consider the following graph $G = (V, E)$.

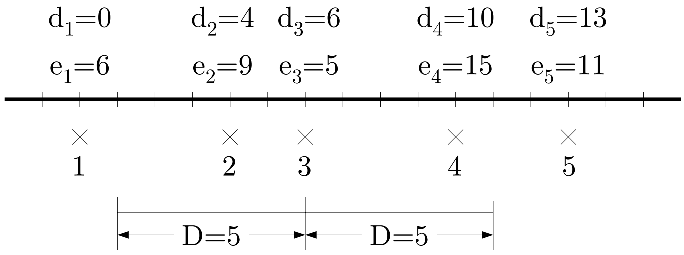


a) Perform a depth-first search (DFS) and a breadth-first search (BFS) on G starting from the vertex A. If more than one possible successor exists, use the first vertex in ascending alphabetic order.

b) What are the running times of BFS and DFS on a graph stored using adjacency lists and not using an adjacency matrix?

Exercise 8.3 *Dynamic Programming: Placing Wind Turbines (Exam Question January 2013).*

We want to place wind turbines along a road to produce energy. Due to geographical reasons n different positions are possible, but laws prescribe that the distance between two wind turbines has to be at least D. The possible positions $d_1, ..., d_n$ are given as coordinates on a line, where the leftmost position has the coordinate value 0. The distance between the i-th possible position and the first possible position is d_i. Hence, we have $d_1 = 0$ and $d_i < d_{i+1}$ for each $i \in \{1, ..., n-1\}$. When a wind turbine is installed at position i, it produces energy $e_i > 0$. The task is to find a positioning of wind turbines that maximizes the total energy yield.



Example: The image above shows a situation for $n = 5$ possible positions. For example, if a wind turbine is installed at position 3, then no wind turbines can be installed at the positions 2 or 4. When the wind turbines are placed on the positions 1, 3 and 5, then they produce $6+5+11 = 22$ units of energy. This solution is not optimal: An installation of wind turbines on the positions 2 and 4 produces $9 + 15 = 24$ units of energy.

a) Provide an example (as simple as possible) that shows that the following greedy strategy does not necessarily lead to an optimal solution: *"Select a possible position with maximal energy yield until no other wind turbines can be placed."*

b) Describe a dynamic programming algorithm that computes the maximal producible energy as efficiently as possible.

c) Specify the running time of your solution.

d) Describe in detail how the algorithm from (b) has to be modified to additionally compute an optimal placement of wind turbines that produces a maximum amount of energy.

Exercise 8.4 *Cycle detection (**Programming Exercise**).*

In this exercise we are going to perform cycle detection on undirected graphs. We are given a connected undirected graph G represented by adjacency lists. A cycle is a walk in G that starts and ends in the same vertex. Here, the graph G has n vertices and the vertices are labeled by the numbers $0, \ldots, n-1$. Develop an algorithm that detects if there is a cycle in graph G.

Input The first line contains only the number t of test instances. The first line of each instance contains n the number of vertices of the graph. Then, we have another n lines that describe the adjacency list of the graph. That means, the ith line contains all the vertices that are adjacent to the vertex with label $i-1$.

Output For every test instance we output only one line. This line contains the character `y` if the graph G contains a cycle and `n` otherwise.

Example

Input:

```
2
3
1
0 2
1
3
1 2
0 2
1 0
```

Output:

```
n
y
```

Directions We provide you with a template. It contains the necessary code to read the input. The graph is then represented by its adjacency lists in `ArrayList<Integer>[] graph`, which is an array (of length n) of ArrayLists of integers. Each ArrayList represents the adjacency list of the corresponding vertex.

There is only one testset for 100 point in this exercise.

Hand-in: Wednesday, 27th April 2016 in your exercise group.