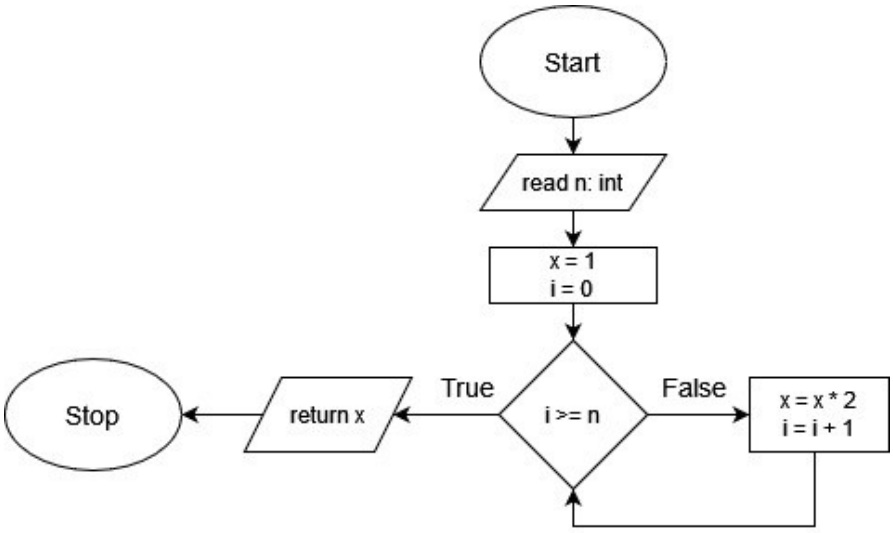
Universität Konstanz

Introduction to Computing for the Social Sciences
Exercise Sheet for Session 06

Prof. Dr. David Garcia

Exercise 1: Flow charts

The flowchart below represents the method unknown(n).



a) Based on the flowchart, briefly describe in your own words what the method unknown(n) does.

b) Write down a pseudocode for the method unknown(n)

Exercise 2: Breadth-First Search

Given the following adjacency matrix:

	A	B	C	D	E	F
A	0	1	0	1	0	0
B	1	0	0	0	1	0
C	0	0	0	0	1	1
D	1	0	0	0	0	0
E	0	1	1	0	0	1
F	0	0	1	0	1	0

a) Draw its corresponding graph

b) Write how it can be encoded with adjacency lists

c) Starting from node B, show how Breadth-First Search would process the graph. Indicate the states of the queue during the execution of the algorithm and the resulting distances and parent relationships on the graph. When introducing nodes in the queue, do it in alphabetic order.

Exercise 3: Understanding graph algorithms

Given the following adjacency matrix for a directed, weighted graph:

from \ to	A	B	C	D	E
A	0	6	0	7	0
B	0	0	5	8	-4
C	0	-2	0	0	0
D	0	0	-3	0	9
E	3	0	7	0	0

a) Draw its corresponding graph with the weight of each edge.

Consider the Bellman-Ford algorithm below. It calculates the weighted shortest paths from a source node s in graph G, where G.V are its vertices, G.E are its edges, and w(u,v) is the weight for the edge connecting from node u to node v. Note that weights in this graph can be negative. There is no weight value from node u to node v if there is no edge connecting them.

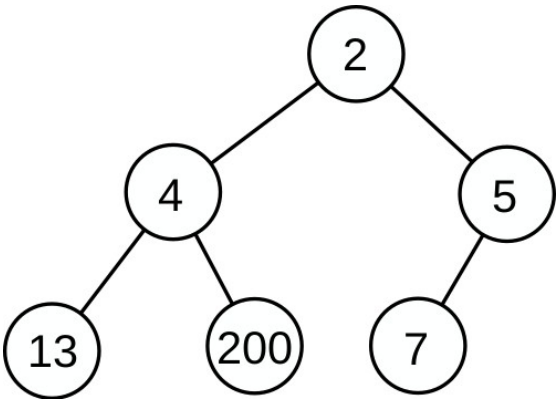
```
1     Bellman-Ford(G, s, w)
2         for each vertex v in G.V:
3             v.distance = INFINITY
4             v.parent = NULL
5         s.distance = 0
6         for i = 1 to |G.V| -1:
7             for each (u,v) in G.E:
8                 if v.distance > u.distance + w(u,v):
9                     v.distance = u.distance + w(u,v)
10                    v.parent = u
```

b) Show the final result of running Bellman-Ford in the graph of question a starting from node A. When iterating over edges in the loop starting on line 7, do it alphabetically with priority on u and then on v (i.e. (x,y) goes right before (x,z) and that right before (y,a)).

c) Consider the case in which the weight of the edge connecting from node C to node B is -4 instead of -2. Describe in your own words how the algorithm would behave and how this influences the result.

Exercise 4: Recursive algorithms

A min-heap is a data structure similar to a binary search tree but simpler. Min-heaps take the form of a nearly-complete binary tree (all levels are full except the last one, which might not be full and is filled from the leftmost leaf) with the condition that each node contains a key that is smaller than or equal to the keys of its children. Below you have an example:



Min-heaps can be implemented in arrays such that levels of the tree are stored sequentially in the array positions between 1 and N, where N is the total size of the min-heap. This way you can calculate the left child, right child, and parent of a node in position x as:

left-child(x) = 2x right-child(x) = 2x+1 parent(x) = floor(x/2)

a) Write the pseudocode of an iterative function called reduce(H,x,y) that takes a min-heap H and reduces the value of the node in position x to value y, assuming that y is lower than the current value stored in position x. The function has to change the min-heap such that the result is still a min-heap (each node contains a key that is smaller than or equal to the keys of its children). Use and manipulate the min-heap H with indices as you would do with an array.

b) Write the recursive version of your code above. Which type of recursion have you implemented?

c) Show how reduce(H,x,y) can be applied to the min-heap of the example above to reduce the value of the last node (the one that contains 7) to the value 1. Indicate any swaps or value changes and the final state of the min-heap.

Exercise 5: Depth-First Search

Given the following adjacency matrix:

	A	B	C	D	E	F
A	**0**	**1**	**1**	**0**	**0**	**0**
B	**1**	**0**	**1**	**0**	**0**	**0**
C	**1**	**1**	**0**	**1**	**1**	**0**
D	**0**	**0**	**1**	**0**	**0**	**1**
E	**0**	**0**	**1**	**0**	**0**	**1**
F	**0**	**0**	**0**	**1**	**1**	**0**

a) Draw its corresponding graph

b) Starting from node B, show the result of running Depth-First Search on the graph. Indicate the resulting discovery and finished times for each node and parent relationships on the graph. When processing the neighbors of a node, do it in alphabetic order.