Algorithms and Data Structures

Assignment 2 – Spring 2018

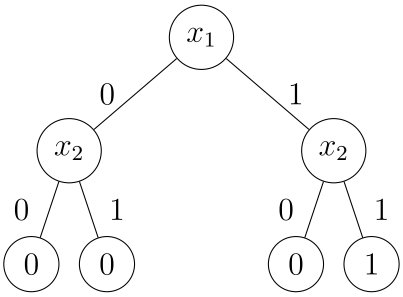
Max Cattafi and Christos Bouganis

Context and general requirements

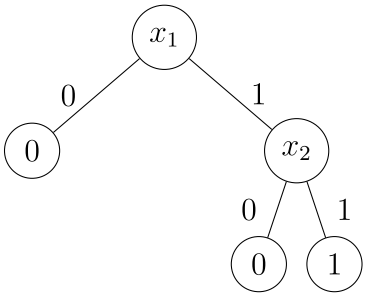
Consider again the “logic and” boolean function:

x_1	x_2	x_1 `and` x_2
0	0	0
0	1	0
1	0	0
1	1	1

Which can be represented as a graph as follows:



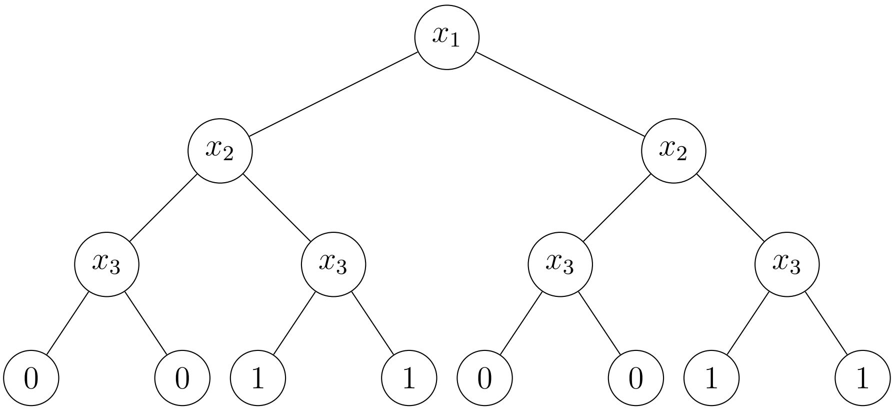
Notice that it could also be represented in an equivalent and more compact way as follows:



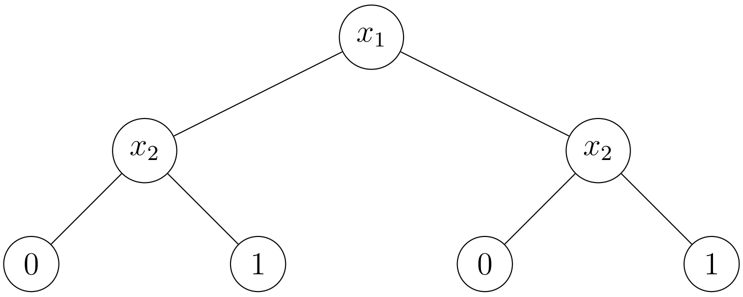
Consider also the following example:

x_1	x_2	x_3	f(x_1, x_2, x_3)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

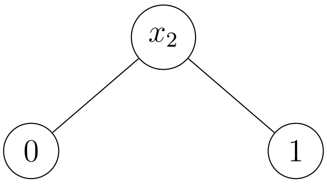
This function can be represented as follows (from now on we omit the annotation for the graph edges, it is intended to follow the usual convention):



A more compact way could be as follows:



However if we are not constrained by the order of the variables in the data structure, we can obtain an even more compact one:



Write an implementation for the following functions:

- `bdt buildcompactbdt(const std::vector<std::string>& fvalues);`

  Builds a compact data structure (there is no constraint on the order of the variables).

- `std::string evalcompactbdt(bdt t, const std::string& input);`

  Evaluates a function represented as a compact data structure (no matter the order of the variables).