Multilevel Tree-Based Lookup Table for Acceleration of Numerical Calculations

Gleb Olegovich Kosheev[1,2]   Ján Buša Jr.[2,3]

[1]Institute of System Analysis and Management, Dubna University, Russia
[2]Laboratory of Information Technologies, JINR, Dubna, Russia
[3]Institute of Experimental Physics, SAS, Košice, Slovakia

AYSS-2020, Dubna
11.11.2020

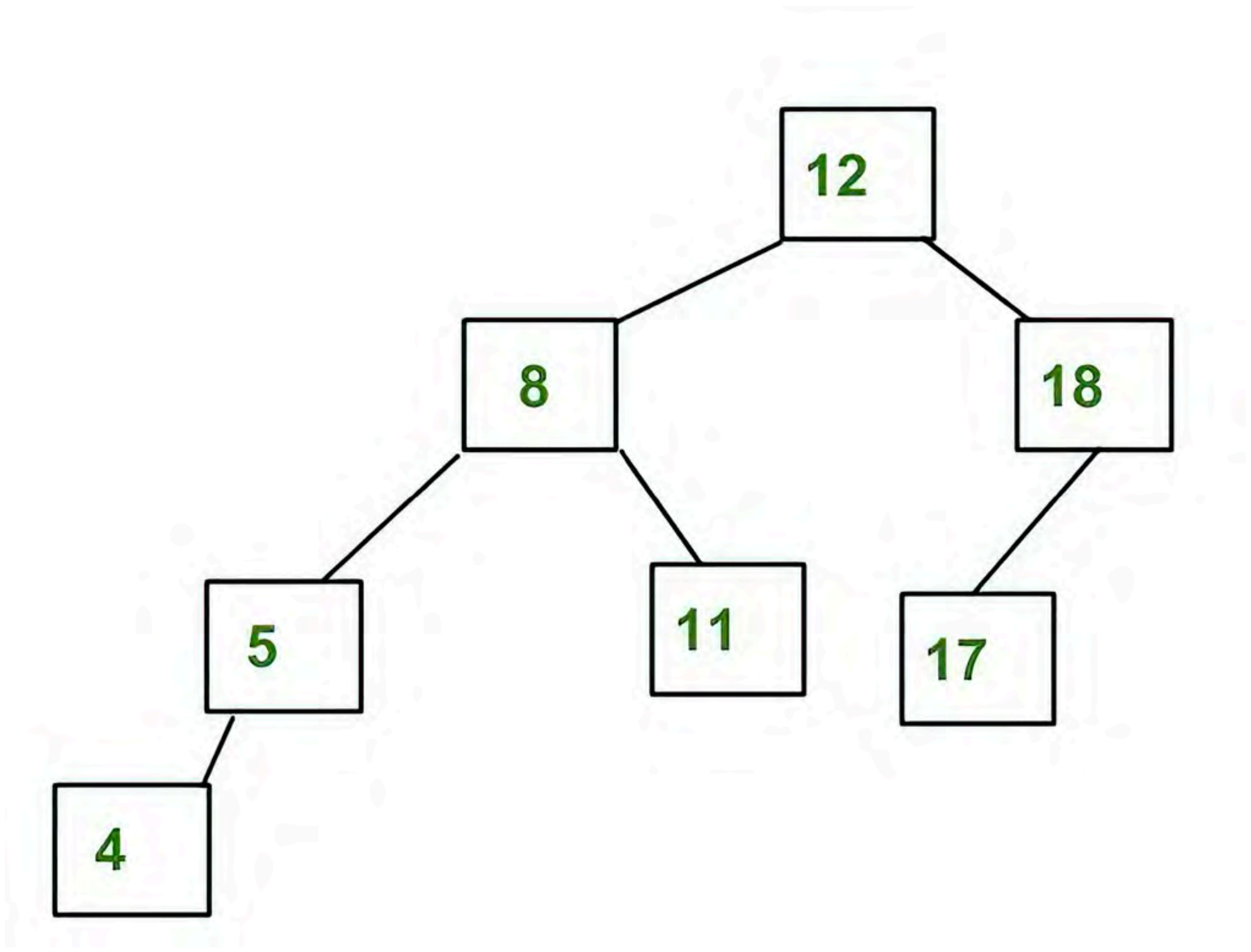
Problem Formulation

- We have program that calls some computationally intensive function many times for the same arguments

- Propose the solution for saving and reusing intermediate results of this function
- Implement proposed solution using C++
- Compare time of proposed method to calculations without cashing

AVL Tree

- Search: O(log *N*)
- Insert: O(log *N*)



Function $f_1(k)$

k – function argument

- Implement AVL tree
- Replace calculation in f_1 *by search from AVL tree*

Function $f_2(k_1,k_2,k_3,\dots,k_{m-1},k_m)$

k – array of arguments

m – amount of arguments of function

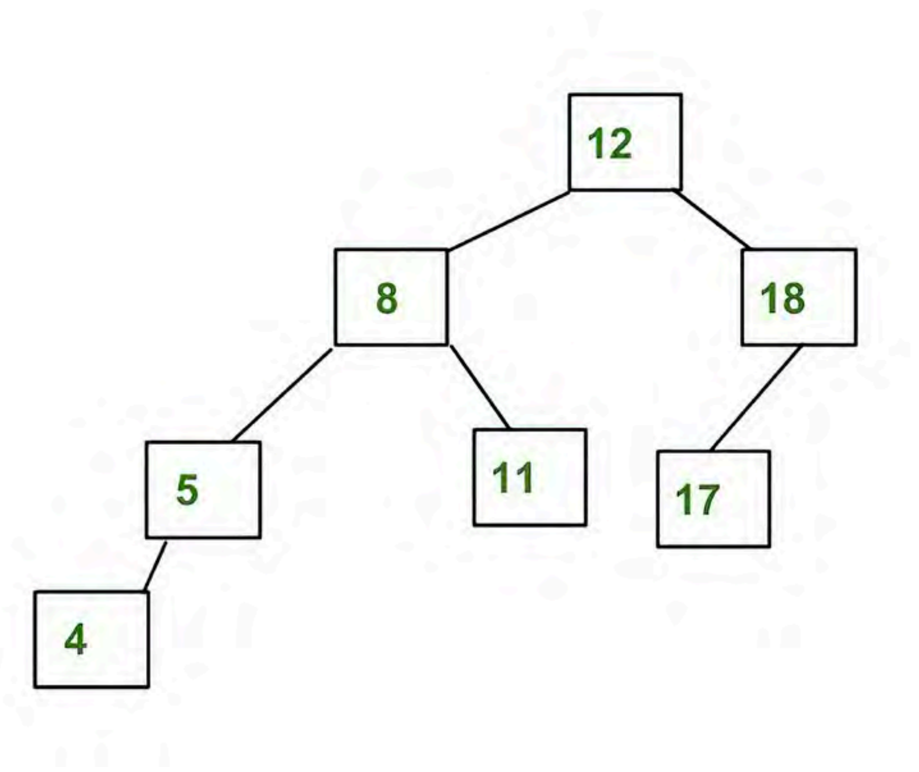
Our structure (AVL's tree leaf) will contain several variables:
data – stores argument

result – stores result of function

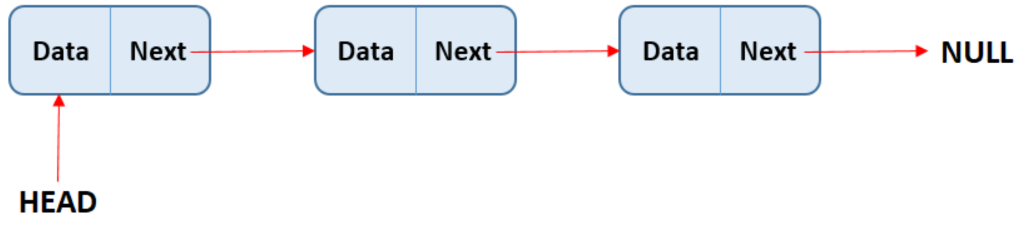
left, right – pointers to leaves

AVL Tree + Linked List =
= Linked List of Trees (LLT)

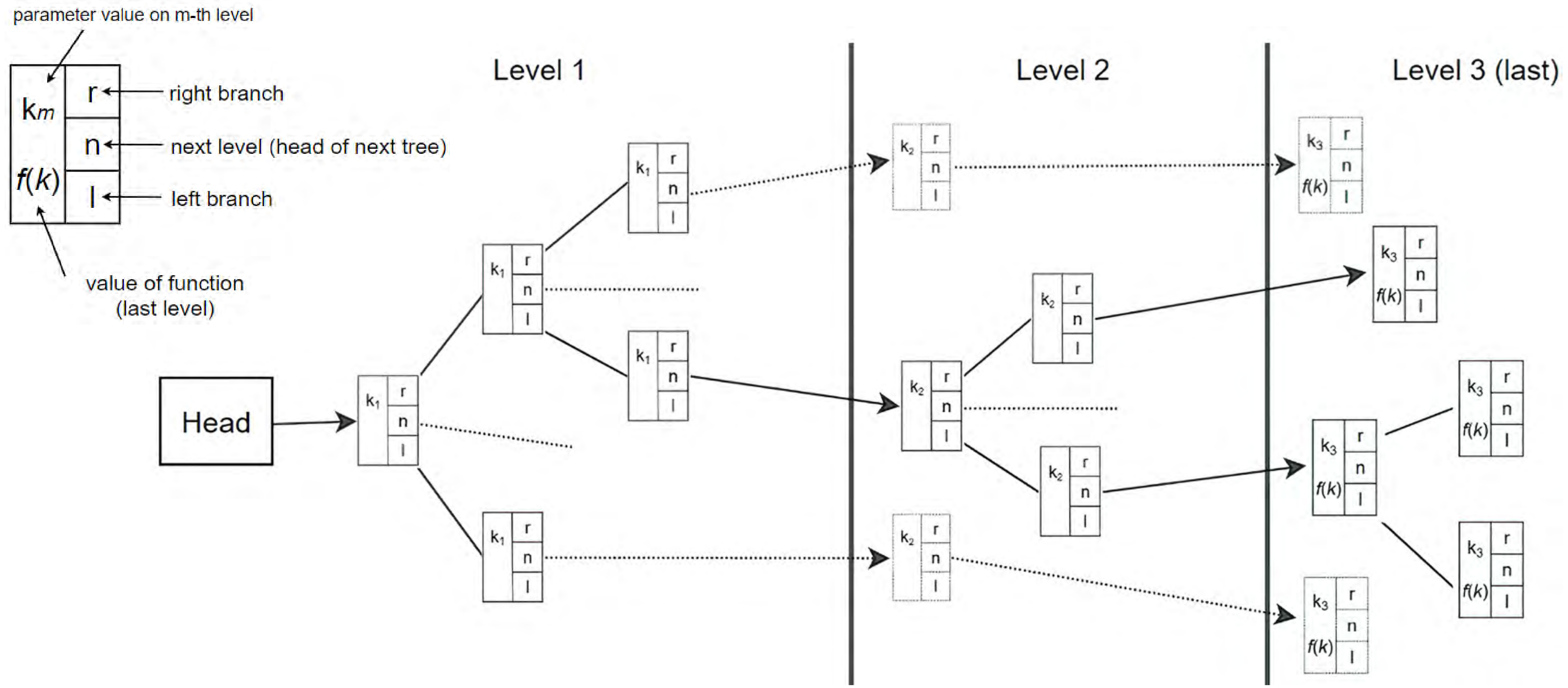
AVL Tree



Linked List



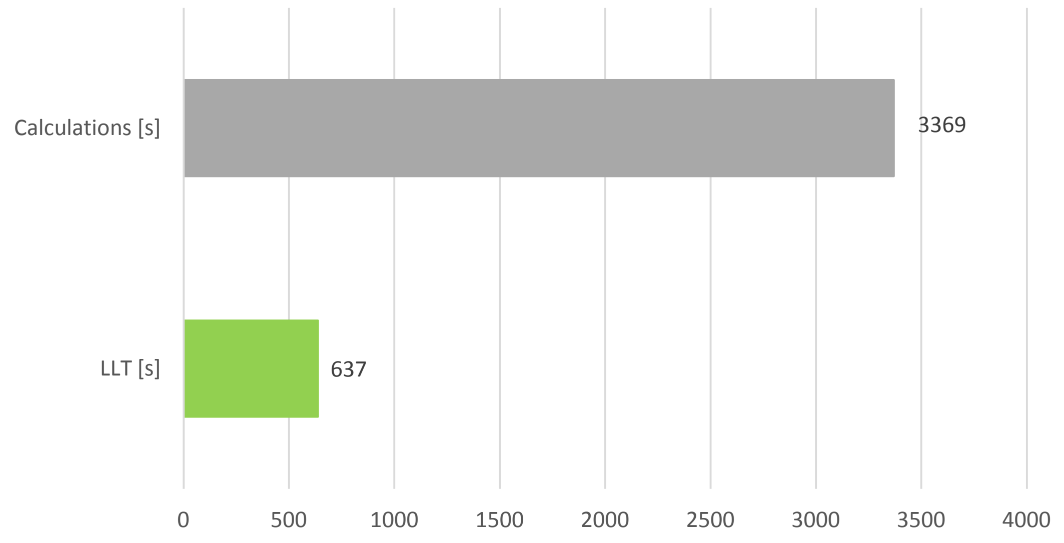
Detailed Example of LLT for Function of m=3 Parameters

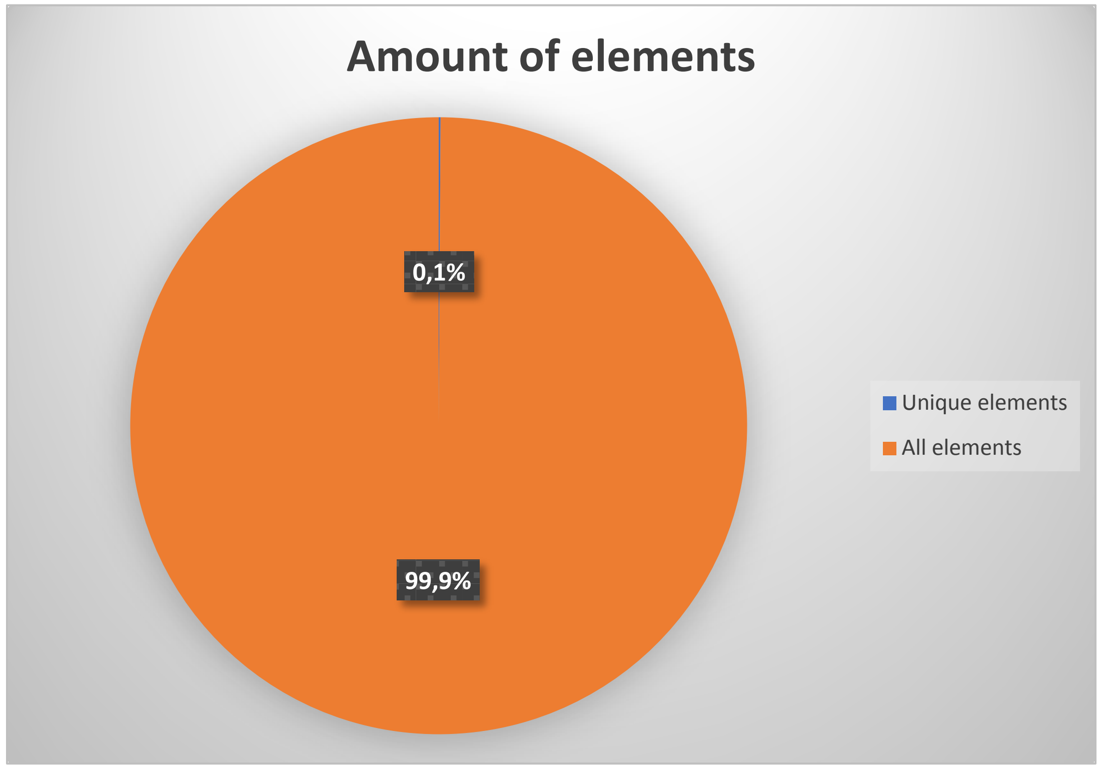


parameter value on m-th level
k_m
$f(k)$
r
right branch
n
next level (head of next tree)
l
left branch
value of function (last level)
Level 1
Level 2
Level 3 (last)
Head
k_1
k_2
k_3

Insert: O(*m* log *N*) Search: O(*m* log *N*)

Performance Comparison





- Calculations: 56 minutes
- LLT: 10.5 minutes (81.1% faster)

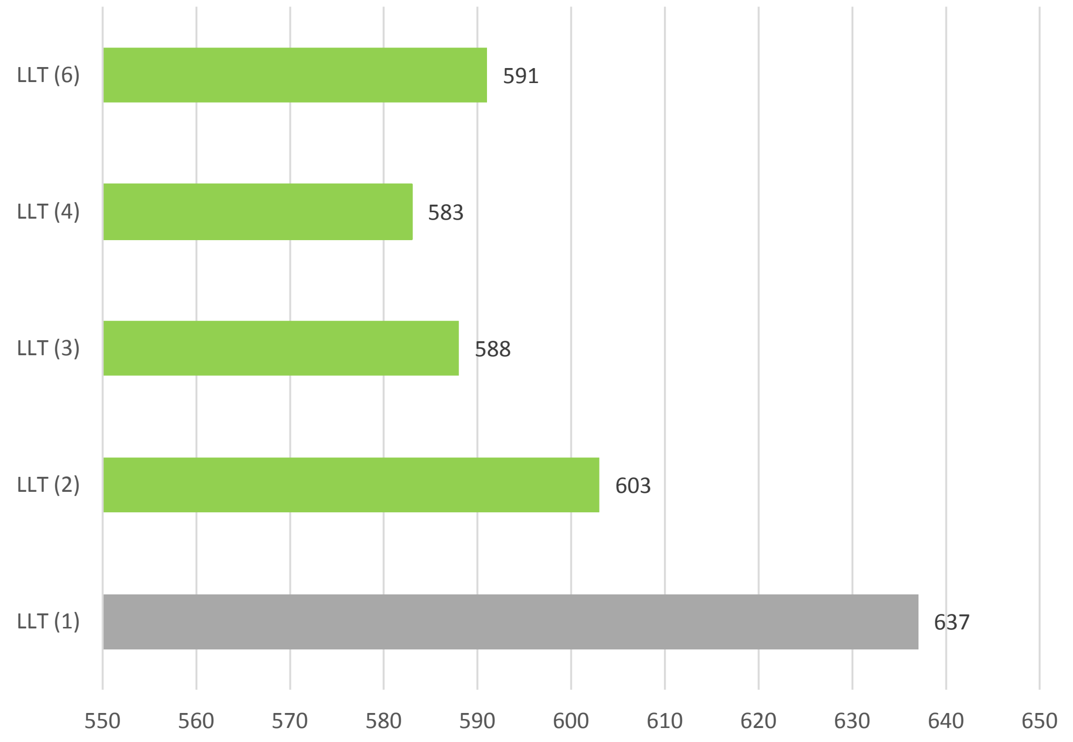
Calculations: GNU FORTRAN
LLT: GNU FORTRAN and g++

Multithreaded Search with OpenMP

Time acceleration (compared to LLT)

- LLT (2) – 5.5%
- LLT (3) – 7.7%
- **LLT (4) – 8.5%**
- LLT (6) – 7.3%

LLT (amount of threads)


LLT (6)
LLT (4)
LLT (3)
LLT (2)
LLT (1)