Data structure inference based on source code

Aleksander Balicki

August 28, 2011

Abstract

1 Introduction

1.1 Example imperative language

We define an imperative language, on which we will show examples of the algorithm.

2 Data structure inference

2.1 Comparison of the complexities

Asymptotical complexity of an operation we store as a pair of type:

$$AsymptoticalComplexity = Int \times Int, \tag{1}$$

where

$$(k,\ l)\ means\ O(n^k \log^l n). \tag{2}$$

The reason to choose such a type is that it's easier to compare than the general case (we can do a lexicographical comparison of the two numbers) and it distincts most of the data structure operation complexities.

Sometimes we have to use some qualified complexities:

$$ComplexityType = \{Normal,\ Amortized,\ Amortized\ Expected,\ Expected\} \tag{3}$$

The overall complexity can be seen as a type:

$$Complexity = AsymptoticalComplexity \times ComplexityType \tag{4}$$

Here we can also use a lexicographical comparison, but we have to say that

$$Normal > Amortized, \tag{5}$$
$$Amortized > Expected, \tag{6}$$
$$Expected > Amortized\ Expected, \tag{7}$$
$$\tag{8}$$

and that $>$ is transitive.

We also always choose the smallest asymptotic-complexity-wise complexity. For example, we have a search operation on a splay tree. It's $O(n)$, but $O(\log n)$ amortized, so it's represented as $((0, 1), Amortized)$.

2.2 Comparison of the data structures

We define a set *DataStructureOperations*. We can further extend this set, but for now assume that

$$DataStructureOperations = \{Insert,\ Update,\ Delete,\ FindMax,\ DeleteMax,\ \ldots\}. \tag{9}$$

Each of the *DataStructureOperations* elements symbolizes an operation you can accomplish on a data structure.

The type

$$DataStructure \subset DataStructureOperations \times Complexity \tag{10}$$

represents a data structure and all of the implemented operations for it, with their complexities.

When trying to find the best suited data structure for a given program P, we look for data structure uses in P. Let $DSU(P)$ be the set of *DataStructureOperations* elements, that are used somewhere in the source code of P.

2.3 Choosing the best data structure

3 Extensions of the idea

3.1 Second extremal element

3.2 Big load

change in the algorithm

3.3 Data structure modifications

max elem cache

3.4 Linked data structures

keeping records

3.5 Transforming datastructures on-line

what it said

3.6 Upper bound on the element count

so we can choose between malloc and static allocation

3.7 Outer-world input

detecting scanf and sockets and so on

4 Program

4.1 Recommendation mode

prints recommendations

4.2 Advice mode

prints advice

4.3 Compile mode

linkes appropriate lib

4.4 Typechecker