Analysis of the Expected Search Cost in Skip Lists*

Thomas Papadakis                J. Ian Munro
Department of Computer Science
University of Waterloo
Waterloo, Ont N2L 3G1
Canada

Patricio V. Poblete
Departamento de Ciencias de la Computación
Universidad de Chile
Casilla 2777
Santiago, Chile

Abstract

Skip lists, introduced by W. Pugh, provide an alternative to search trees. The exact value of the expected search cost is derived (in terms of previously studied functions), as well as an asymptotic expression for it. The latter suggests that Pugh's upper bound is fairly tight for the interesting cases.

1 Introduction

The skip list, recently introduced by W. Pugh [5], is an interesting and practical alternative to search trees. The approach is to store each of its \( n \) \( (n \geq 1) \) elements in one or more of a set of sorted linear linked lists. All elements are stored in sorted order on a linked list denoted as level 1, and each element on the list at level \( i \) \( (i = 1, 2, \ldots) \) is included with (independent) probability \( p \) in the list at level \( i + 1 \). A header contains the references to the first element in each list (see Figure 1). The height of the data structure, that is, the number of linked lists, is also stored.

A search for an element begins at the header of the highest numbered list. This list is scanned, until it is observed that its next element is greater than or equal to the one sought (or the reference is null). At that point, the search continues one level below until it terminates at level 1 (see the search for the 6th element in Figure 1). We have adopted

---

*This research was supported in part by the Natural Science and Engineering Research Council of Canada under grant No. A-8237, the Information Technology Research Centre of Ontario, and FONDECYT (Chile) under grant 90-1263.
the convention that an equality test is done only at level 1 as the last comparison. This is
the usual choice in a standard binary search and avoids two tests (or a three way branch)
at each step.

The search cost is defined as the number of pointer inspections excluding the last
one for the equality test. For the search in Figure 1, this is $9.1$. Of principal interest
is $C_p(m, n)$ ($m = 1, 2, \ldots, n + 1$), which, assuming that the 0th element is $-\infty$ and the
$(n+1)$st element is $+\infty$, is defined as the average search cost when performing a successful
search for the $m$th element, or an unsuccessful search for an element between the $(m-1)$st
and the $m$th in a list of $n$ elements. This average is taken over all possible skip lists, for
fixed $m$ and $n$.

It is worthy of note at this early stage that the expected search cost will depend on
the relation between the element sought and the other elements in the skip list. The
expected cost of a search for the element that turns out to be the $m$th smallest, will be
an increasing function of $m$. In particular, searching for the smallest element, or one less
than the smallest, will have cost equal to the height of the structure, while searching for
other elements will have the additional cost of following links horizontally. It is clear that
the height of a skip list is a lower bound on the (expected) search cost for any element.
The spirit of the structure is that $p$ be a probability of moderate size, say $\frac{1}{2}$ or $\frac{1}{3}$.

Insertions and deletions (as defined by Pugh) are very straightforward. A new element
is inserted where a search for it terminated at level 1. As it is put in list $i$ ($i = 1, 2, \ldots$),
it is inserted, with probability $p$ ($0 < p < 1$), where its search terminated at level $i + 1$.
This continues until, with probability $q = 1 - p$, the choice is not to insert. The counter
for the height of the data structure is increased, if necessary. Clearly, this suggests the
notion of nodes with $i$ pointer fields and a data field. Deletions are completely analogous
to insertions. An element to be deleted is removed from the lists in which it is found. The
height of the data structure is updated by scanning the header’s pointers by decreasing
level until a non-null pointer is found.

At this point we can make an observation on the adopted “scan the list until the next
list element is greater than or equal to the one sought” criterion to drop down a level
during the search. Had it been simply “greater than the one sought”, then the search
path for the $m$th element in this case would have been identical to the search path for

---

1Note that our definition of the search cost is 1 greater than Pugh’s definition. Our references to his
results are translated into our terms.