[Skip List与Hierarchical Navigable Small World 2](#_Toc738001207)

[简介 2](#_Toc1868224624)

[Skip List 2](#_Toc892852781)

[HNSW 4](#_Toc1708448678)

[结束语 5](#_Toc2040570756)

[参考 5](#_Toc558853502)

# Skip List与Hierarchical Navigable Small World

## 简介

Skip List，即跳表是平衡树的一种。它在概率上实现平衡，而不是严格平衡，因此插入和删除操作非常简单。它的多层结果是随机方式生成的，不需要重平衡的过程。对比平衡树和其他自我调整平衡的树结构，跳表的速度性能更高，存储空间使用更高效。跳表易于实现、扩展和修改，适合替代应用平衡树的场景[1]。

在跳表之后，出现了各种“跳”式数据结构，如跳树[2][3]，跳图[4]，跳树图[5]，跳网[6]。这里不做过多说明。

Hierarchical Navigable Small World(HNSW[7])是在高维数据检索应用中著名的近似近邻查找算法。其同样借鉴跳表生成多层的方法，生成多层图结构，实现高效的索引生成与检索[8]。

下面对跳表和HNSW分别讨论。

## **Skip List**

**直接用图展示一下跳表数据结构。查找的流程如图1所示：**

**查找52**

header

tail

32

32

32

32

32

32

30

30

30

21

8

8

88

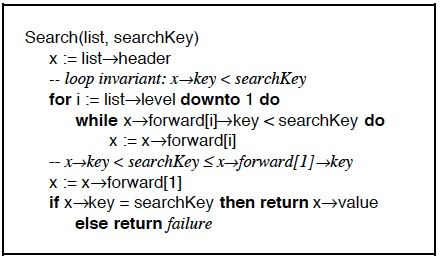
88

71

52

1. **查找流程**

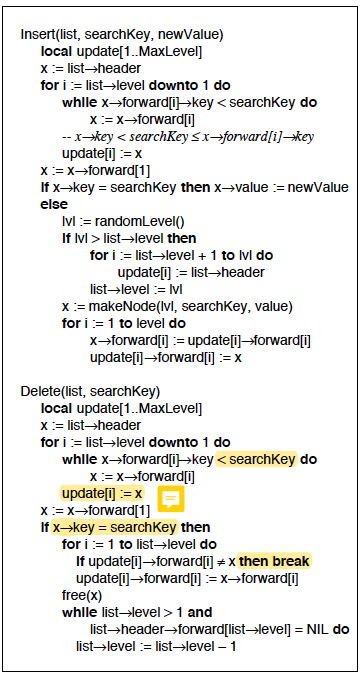
**我们从论文中截一下算法部分，如图2所示，需要强调的是在查找，插入和更新流程中，执行相同的查找步骤，但是在修改流程中，需要记录下key项在每层中的前置项作为后续修改需要更新的数据。**

****

1. **查找**

**如果是插入数据，则先执行查找流程。如果key已经存在，则只更新key对应的value。如果是新的key，则确定level后，则更新所有前置项中forward指向新key项，新key项的forward则指向前置项的原forward值。**

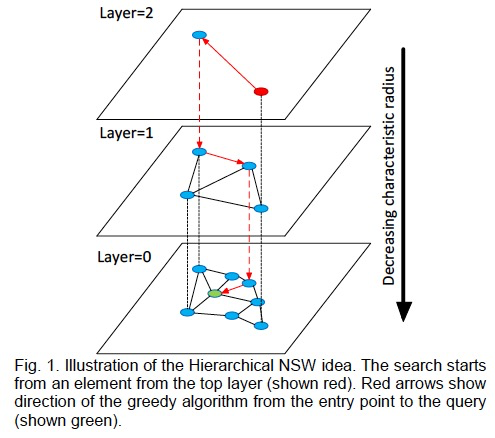
**删除时，将key项的前置项的forward值设为key项的forward值。同时要修改跳表的level值。**

****

1. **插入与删除**

## **HNSW**

**接下来讲一下HNSW，HNSW借鉴跳表的思想，将一个二维的单层图结构扩展为多层图结构。如图4所示，HNSW主要应用于多媒体等高维数据的检索，具体请参考论文[7]。**

****

1. **HNSW**

## **结束语**

前几个小节介绍了跳表和HNSW。通过两个算法的介绍我们可以看到，在我们新的应用中，我们可以通过借鉴前人的思想来解决新的问题。

## **参考**

1. **<http://homepage.cs.uiowa.edu/~ghosh/skip.pdf>**
2. **<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.514>**
3. **<http://www.cs.yale.edu/homes/aspnes/papers/opodis2005-b-trees-final.pdf>**
4. **<http://www.cs.yale.edu/homes/aspnes/papers/skip-graphs-journal.pdf>**
5. **<http://www0.cs.ucl.ac.uk/staff/a.gonzalezbeltran/pubs/icc2007.pdf>**
6. **<https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2002-92.pdf>**
7. **<https://arxiv.org/abs/1603.09320>**
8. **<http://yongyuan.name/blog/opq-and-hnsw.html>**