# Lecture Notes on *Data Structures*

M1522.000900



Seoul National University

Fall 2022



# Part X

# Algorithm Designs

## Greedy Algorithms

- "Take the best among those available NOW" approach.
- Hope that obtaining a series of *local optimum* solutions leads to the *global optimum*.
- Example: Huffman coding.
  - Take the most infrequent code first so that it is assigned to the deepest leaf node.
- Example: Dijkstra's shortest path algorithm.
  1. Choose the best (*i.e.*, shortest distance) among the fringe edges.
  2. This greedy approach (*i.e.*, looking only for local optimum) may not work (*i.e.*, may not yield the global optimum) if there are edges with a negative weight.
- More examples: Prim's MST, Kruskal's MST.
- Greedy algorithms are efficient but need proving the correctness.



## Divide and Conquer

- Divide a problem into two or more smaller subproblems.
- Solve the smaller problems recursively.
- Examples: binary search, quicksort, mergesort, etc.
- Running time of a divide-and-conquer algorithm can be modeled by a recurrence relation.

## Dynamic Programming

Similar to Divide-and-Conquer, but is applicable to more complex problems and relies on iterations instead of recursions.

- Subproblems are often not a clear division but *overlapping* ones of the original problem.
- For a problem formulated by recursive expressions, its recursion is translated into an iterative algorithm by systematically recording the answers to sub-problems in a table.
- Example: Floyd's All-Pair shortest path algorithm.
  1. The computation of shortest k-paths is expressed recursively.
  2. Floyd's algorithm is iterative by recording them in a matrix.
- Example: Longest Common Subsequence, Optimal Binary Search Tree.



## Randomized Algorithm

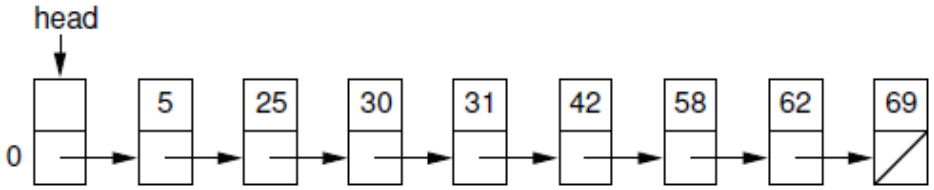
An algorithm is *randomized* if its behavior is determined not only by the input but also by values produced by a random number generator.

- The runtime is not deterministic.
- The main advantage is the simplicity and efficiency in data structures and methods. No particular input elicits its worst case behavior.
- Example: randomized quicksort uses a randomly selected element as a pivot. If this is run twice for the same input array, it is very unlikely to take $\mathcal{O}(n^2)$ time twice.
- Example: (Dynamic) Skip Lists.

# Skip List [Pugh 1989]

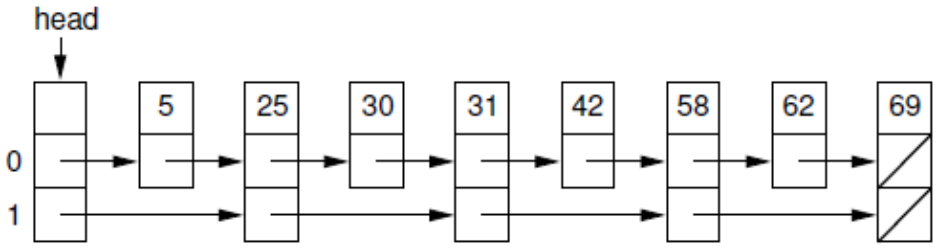
The skip list makes random choices in arranging the keys in such a way that search and insertion times are logarithmic on average.

- Keys are stored in a linked list in sorted order.




  Search and insertion times are $\mathcal{O}(n)$ (*i.e.*, $n$ in the worst case).

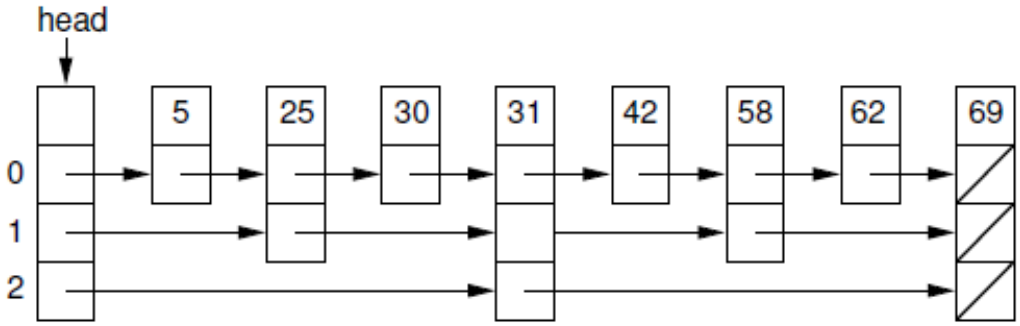
- Additional layer of pointers at every other node.




  Search and insertion times are $\lceil n/2 \rceil + 1$ in the worst case.



- Another layer of pointers at every 4th node.




  Search and insertion times are $\lceil n/4 \rceil + 2$ in the worst case.

- In general, the number of layers is $\log n$, and search and insertion times are $\mathcal{O}(\log n)$.
- The search time is as good as AVL, and simpler than AVL.

# Skip List: Search

**Algorithm 1 (Skip List Search)**

```
Search(k)                                          // k:  search key
   p = head pointer of the topmost layer;
   while(p in one of O(log n) layers) {
      while(key(p.next) < k) p = p.next;  // advance horizontally
      p = p.down;                          // move down to next level
   }
   p = p.next;                                  // move to actual record
   if (key(p) = k) return found;
   else return notfound;
```

This algorithm follows more common convention which refers to the sparsest layer as the topmost one.



# Structure of (Static) Skip List

Review the structure of a (static) skip list.

- The node storing the $m^{th}$ smallest key has $(i+1)$ links if $m$ is a multiple of $2^i$ but not a multiple of $2^{i+1}$.
- Insertions and deletions make it difficult to follow the rule.
- The links are distributed across the nodes as follows.
  - The number of nodes with one link is $n/2$.
  - The number of nodes with two links is $n/4$.
  - ...
  - The number of nodes with $k$ links is $n/2^k$.

  In other words, the probability of a node having $k$ links follows an exponential distribution (*i.e.*, $f(k) = 2^{-k}$).

# (Dynamic) Skip List

Relax the structural restrictions to allow insertions and deletions. In particular, give up the first property in the previous slide, and mimic it by adopting the same probability distribution.

To insert a new key,

1. Determine the number of links (dynamically at insertion time) following the exponential distribution: $prob[k] = 2^{-k}$.
2. Adjust the number of links in the head node if necessary.
3. Keep track of the nodes and links where *level drops* were done while searching.

The exponential distribution can be simulated by flipping a coin. Keep flipping a coin until the head comes up. When the head comes up, the total number of flips made till then becomes the value of $k$.



## Example 1

Insert five keys 10 20 5 2 30 in the given order into an empty skip list.