Lecture 13

Skip Lists

Self Organizing Lists

October 07, 2021
Thursday

LINKED LIST | ISSUES

- The search starts from the beginning of the list and stops
    - EIther item is found.
    - Or list is empty
- Searching can be speed up if the items are ordered.
    - However, a sequential search is still required.

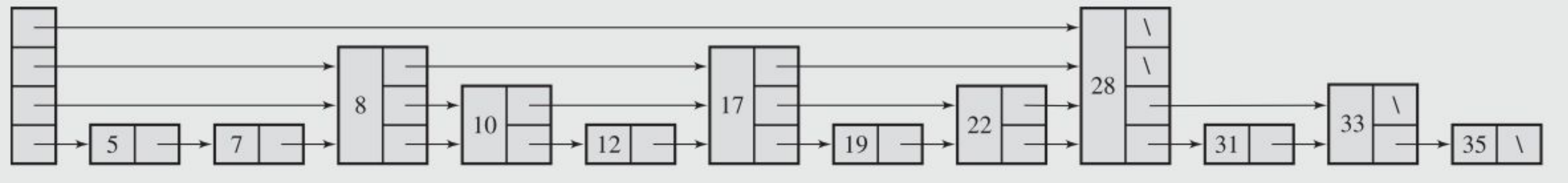
What If,

we can avoid some nodes in our search..!

SKIP LIST

- Skip list is an interesting variant of the ordered linked list that makes such a non-sequential search possible (Pugh 1990).
- In a skip list of ***n*** nodes
  - for each ***k*** and ***i*** such that
  - 1 <= k <= [lg n]
  - 1 <= i <= [n/2^{k-1}] - 1
  - The node (2^{k-1} . i) points to the (2^{k-1} . (i + 1))
  - Every second node points to the node two positions ahead.
  - Every fourth nodes points to the node four positions ahead.

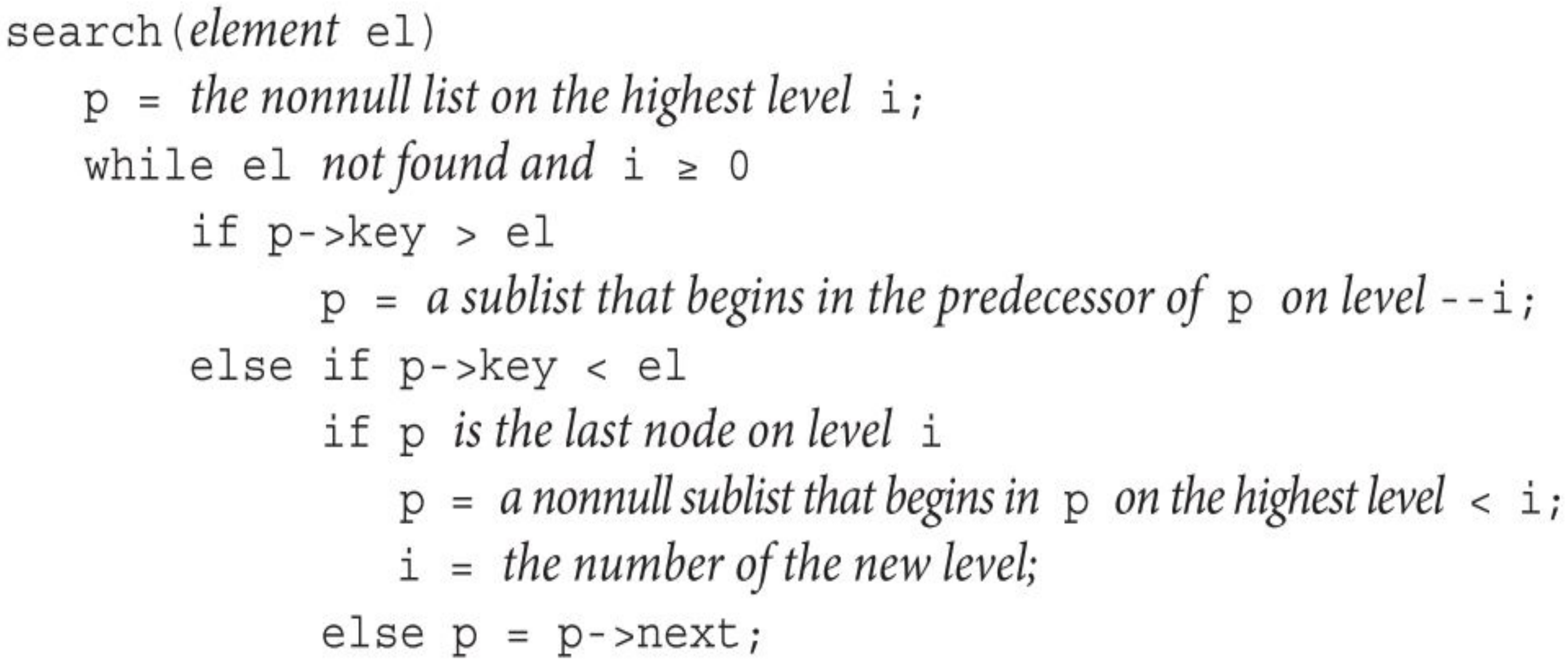
SAMPLE SKIP



SKIP LIST

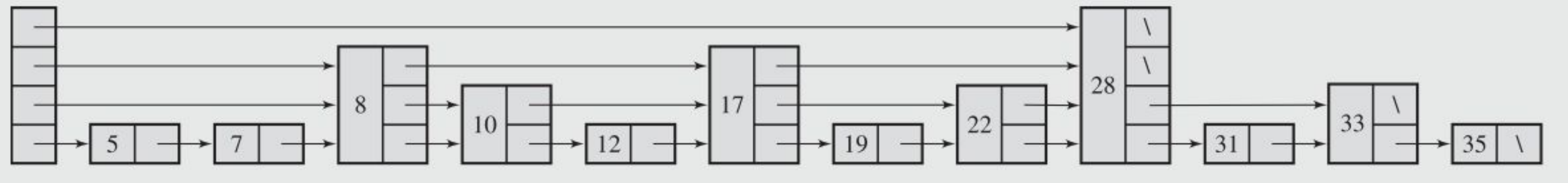
- This requires having different number of pointers in nodes on the list.
  - Half of the nodes have just a single pointer.
  - One-fourth have two pointers.
  - One-eighth have three pointers.
- Number of pointers indicate the level of each node.
  - Numbers of level = [lg n] + 1

SKIP LIST



```
search(element el)
    p = the nonnull list on the highest level i;
    while el not found and i ≥ 0
        if p->key > el
            p = a sublist that begins in the predecessor of p on level --i;
        else if p->key < el
            if p is the last node on level i
                p = a nonnull sublist that begins in p on the highest level < i;
                i = the number of the new level;
            else p = p->next;
```

SEARCH IN SK | FINDING 16



SKIP LIST

1. Level four is tried first.
   a. First node is 28.
2. Level three is tried
   a. First node is 8
   b. Second node is 17.
3. Level two is tried
   a. Starts from Node 8
   b. Then moves to 10 and then again to 17.
4. Level 1 is tried
   a. Starts from Node 10.
   b. Then moves to 12 then to 17.
   c. This is the least level return unsuccessful.

SKIP LIST

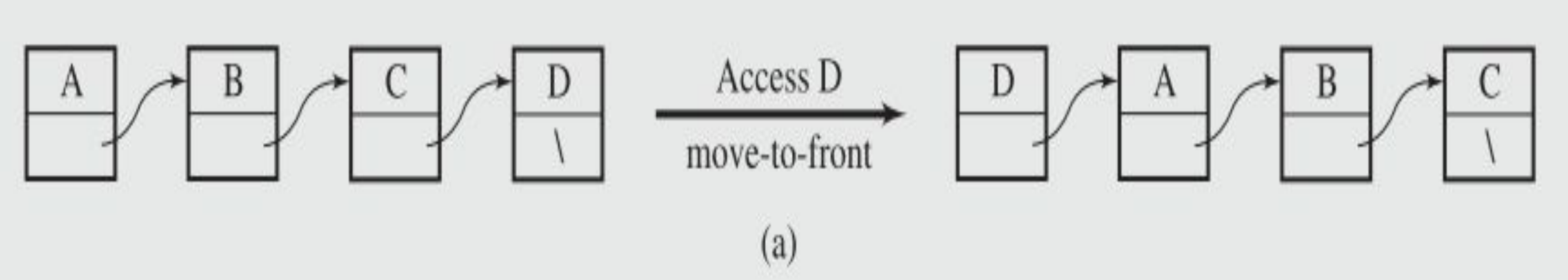
1. Searching is efficient.
2. However the design may lead to very inefficient
   a. Insertion
   b. Deletion
3. All nodes following the node just inserted have to be reconstructed.
4. The number of pointers and the values of pointers have to be changed.
5. For further exploration read 3.4 Skip Lists from Adam Drozdek Book.
6. MIT Lecture on Randomization: Skip List by Srinivas Devdas.

SELF ORGANIZING LIST

1. The motivation for Skip List was to speed up the search process of linked list.
2. Single and Double Linked List require sequential search.
3. Efficiency can be improved by dynamically organizing the list in a certain manner.

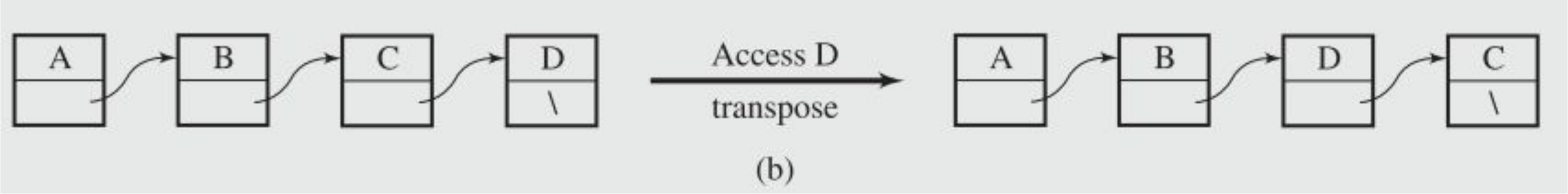
SELF ORGANIZING LIST

1. Move to the Front
   a. After finding the desired node, move it to the front of the list.



SELF ORGANIZING LIST

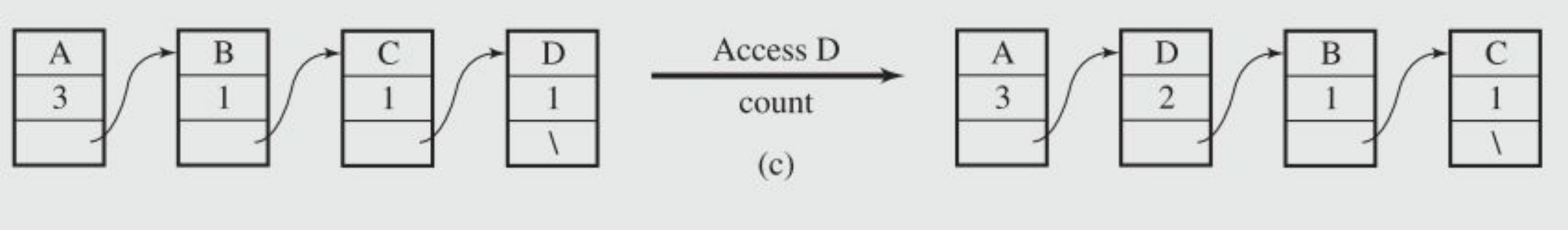
1. Transpose Method
   a. After finding the desired node, swap it with its predecessor.
   b. Unless it is at the head of the list.



(b)

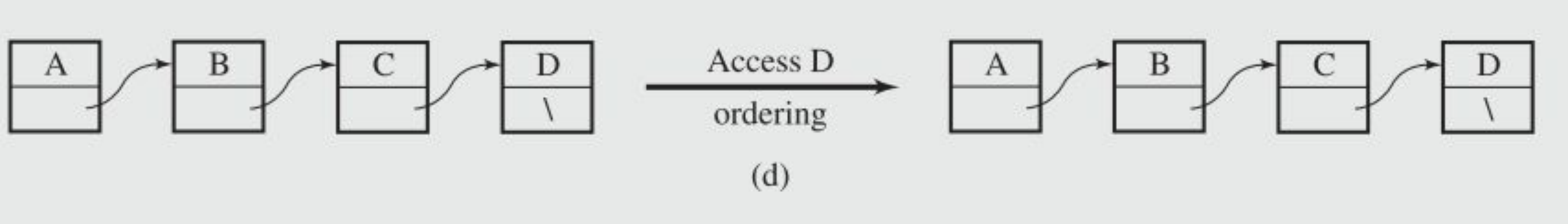
SELF ORGANIZING LIST

1. Count Method
   a. Order the nodes by the number of times they were accessed.



SELF ORGANIZING LIST

1. Ordering Method
   a. Order the list using certain natural criteria for the information



(d)