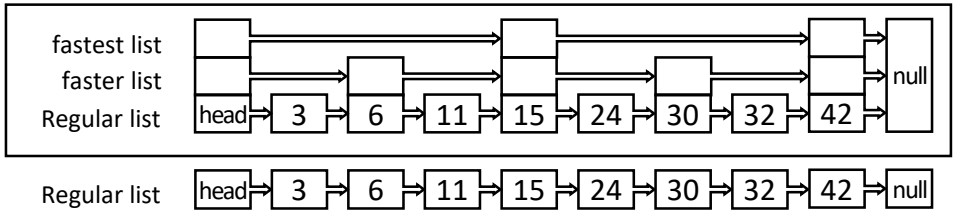
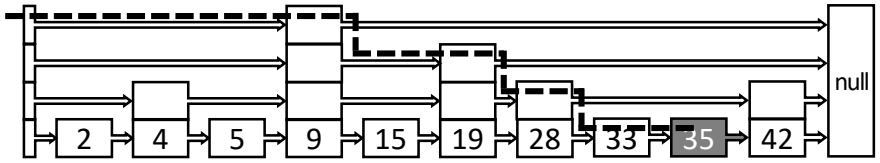
Skip List (Reference: http://www.sable.mcgill.ca/~dbelan2/cs251/skip_lists.html)

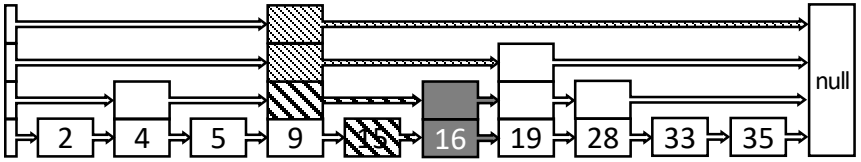
- Terminology
  - A ***forward*** pointer is a pointer that points to a node ahead in the list.
  - A ***level i*** node is a node that has *i* forward pointers.
- Skip List VS. Regular List
  - Trade space for speed



  - If *current* is the node "6", then its level is 2 (size of *current.forwards* being 2);
  - *forwasd0* and *forward1* are "11" and "15".
- Initialization (an empty skip list)
  - Contains only a head whose level is 1 and point it to `null`
- Search for a *value* by *key*
  - Starts from *head*'s top level (fastest list);
  - *forward* is `null`, or *current.key* < *key* ≤ *forward.key*:
    - *forward* not `null` and *forward.key* = *key*: return *forward.value*;
    - Already at lowest level: target not found, return `null`;
    - Otherwise: move down to level-1 (*current* being a turning node);
  - Otherwise: move forward



- Insert a *key-value* pair
  - Search, and maintain a list of pointers – *updates* – containing all the turning nodes;
  - *key* found: update *value* and done, return;
  - *key* not found: create *new node* with **random** level, and point its *forward* in each level to `null` first;
  - From level 0 to min(*current level*, *new level*) - 1:
    - Set *new node*'s *forward* to the *forward* of *updates*;
    - Set *forward* of *update* to the *new node*;
  - If *new level* > *current level*: raise *head*'s level, and point *head*'s new *forwards* to the *new node*.



- Delete a *value* by *key*
  - Similar as insertion – search, and maintain *updates.* Turning nodes as: *current.key* < *key* ≤ *forward.key* (*forward* being `null` is NOT a turning node);
  - *Key* not found: deletion failed, and return `null`;
  - In each level of *updates*:
    - If *forward* is not `null`, then unlink the current node: set *forward* of *update* to the *forward*'s *forward*;
  - Remove levels where *forward* of *head* is `null`;
  - Return *forward*'s *value*.