The Dictionary Principle

•At each iteration, we reduce the size of our problem by a factor of two.

• That means that it takes lg N (i.e. log2 N) steps.

• on average would have taken N/2 steps into a problem that takes lg N steps.

• The JVM is what actually runs your program.

• Its input is called byte-code and it translates this into architecture-specific instructions.

• It knows about: int, double, long, float, char, short, byte, array, reference and how to load classes.

(int) 3.7 is 3. Casting a double to an int truncates toward zero.

• It is required, by API “contract”, that hashCode be consistent with equals such that:

• if a.equals(b) then a.hashCode==b.hashCode

• It also follows that: if a.hashCode != b.hashCode then !a.equals(b)

• Data types: A data type is a set of values and a set of operations on those values.

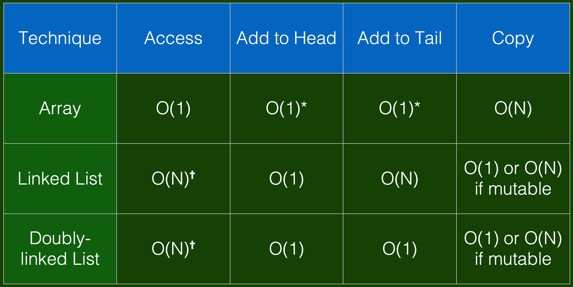
• Abstract data types: An abstract data type is a data type whose internal representation is hidden from the client (encapsulation).

• Objects: An object is an entity that can take on a data-type value. Objects are characterized by three essential properties: The state of an object is a value from its data type; the identity of an object distinguishes one object from another; the behavior of an object is the effect of data-type operations. In Java, a reference is a mechanism for accessing an object.

• Encapsulation.

• A hallmark of object-oriented programming is that it enables us to encapsulate data types within their implementations, to facilitate separate development of clients and data type implementations. Encapsulation enables modular programming.

• Also known as information-hiding.



• A bag is a collection where removing items is not supported—its purpose is to provide clients with the ability to collect items and then to iterate through the collected items (the client can also test if a bag is empty and find its number of items).

•FIFO queues A FIFO queue (or just a queue) is a collection that is based on the first- in-first-out (FIFO) policy.

•A pushdown stack (or just a stack) is a collection that is based on the last-in-first-out (LIFO) policy. (example: Arithmetic expression evaluation)

Linked list:

• Each element has two fields:

• The value of this element; • A pointer/reference to the next element (which may be null).

• Addition/removal of an element: • at the head is O(1), i.e. constant; • at the tail is O(N), i.e. it varies according to the current length N

• A linked list is perfectly suited to a Stack, because all addition/removal operations (push, pop) happen at the head.

Double Linked List:

• Each element has three fields:

• The value of this element;

• A pointer/reference to the next element (which may be null).

• A pointer/reference to the previous element (which may be null).

• Addition/removal of an element: • at the head is O(1), i.e. constant; • at the tail is O(1), i.e. constant;

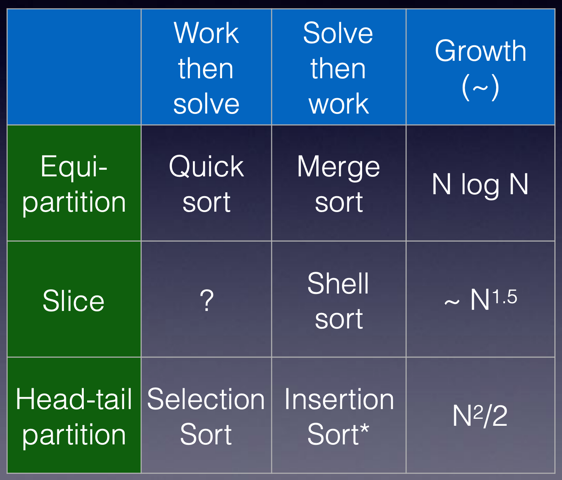
• We’ve shown that in order to comparison-sort an array of N elements, we require at least:

• N log N cams.

• If we use reduction to simplify our problem, there are two choices to be made:

• Equal (or quasi-equal) partitions vs. head/tail partition;

• Work before recursion vs. work after recursion.

****

• An inversion is when two elements of a collection are out of order.

• If a collection of length N is random (i.e. no imposed order), then each element can be compared to (N-1)/2 other elements.

• Such a pair of elements is inverted randomly true or false, so that means on average such a collection has N(N-1)/4 inversions.

• There are 2 forms of swap on an array a of length N:

• swap(a, i)— • this form exchanges elements i and i-1.

• It is a “stable” swap but it can only “fix” one inversion.

• swap(a, i, j)— • this form allows a swap of two non-adjacent elements; • It is an “unstable” swap; • It can “fix” anything up to N-1 inversions (although the average number is N/4).

* Less. Is item v less than w ?

private static boolean less(Comparable v, Comparable w) { return v.compareTo(w) < 0; }

* Exchange. Swap item in array a[] at index i with the one at index j.

private static void swap(Comparable[] a, int i, int j) { Comparable swap = a[i]; a[i] = a[j]; a[j] = swap; }

Selection

int N = a.length; for (int i = 0; i < N; i++) { int min = i; for (int j = i+1; j < N; j++) if (less(a[j], a[min])) min = j; swap(a, i, min); }

* Running time insensitive to input. Quadratic time, even if input is sorted. Data movement is minimal. Linear number of exchanges.

Insertion：

int N = a.length; for (int i = 0; i < N; i++) for (int j = i; j > 0; j--) if (less(a[j], a[j-1])) swap(a, j, j-1); else break;

Proposition. To sort a randomly-ordered array with distinct keys, insertion sort uses ~ ¼ N 2 compares and ~ ¼ N 2 exchanges on average.

Best case. If the array is in ascending order, insertion sort makes N– 1 compares and 0 exchanges.

Worst case. If the array is in descending order (and no duplicates), insertion sort makes ~ ½ N 2 compares and ~ ½ N 2 exchanges.

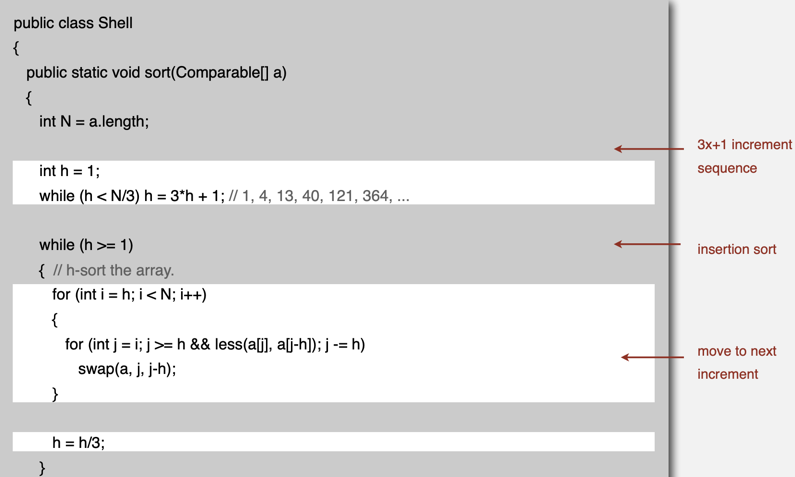
Def. An inversion is a pair of keys that are out of order.

Proposition. For partially-sorted arrays, insertion sort runs in linear time. Pf. Number of exchanges equals the number of inversions (X). Note: Number of compares equals X + (N-1)

Insertion sort: practical improvement

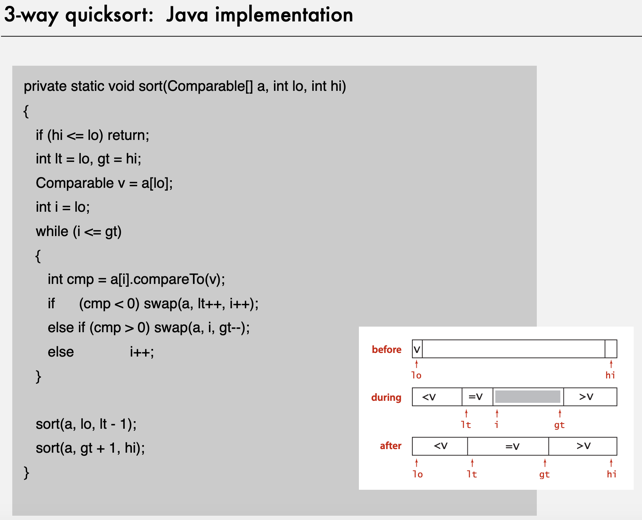
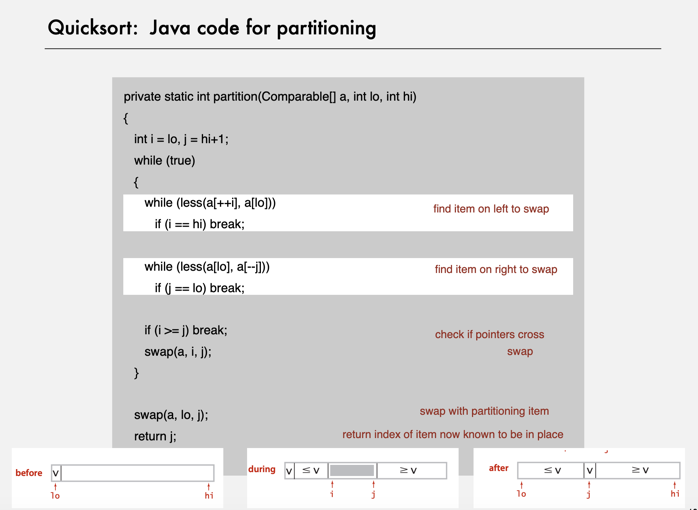
Half exchanges. Shift items over (instead of exchanging). ・Eliminates unnecessary data movement. ・No longer uses only less() and exch() to access data.

Binary insertion sort. Use binary search to find insertion point. ・Number of compares ~ N lg N. ・But still a quadratic number of array accesses.



Why are we interested in shellsort?

Useful in practice. ・Fast unless array size is huge (used for small subarrays). ・Tiny, fixed footprint for code (used in some embedded systems). ・Hardware sort prototype.



Partitioning in-place. Using an extra array makes partitioning easier (and stable), but is not worth the cost. Terminating the loop. Testing whether the pointers cross is trickier than it might seem. Equal keys. When duplicates are present, it is (counter-intuitively) better to stop scans on keys equal to the partitioning item's key. Preserving randomness. Shuffling is needed for performance guarantee. Equivalent alternative. Pick a random partitioning item in each subarray.

Proposition. The average number of compares CN to quicksort an array of N distinct keys is ~ 2N lnN (and the number of exchanges is ~ ⅓ N ln N ).

QuickSort: practical improvement

Median of sample. ・Best choice of pivot item = median. ・Estimate true median by taking median of sample. ・Median-of-3 (random) items.

