Complexity

jok13

I. Informed vs Uninformed Search

A* has lower complexity if it prunes (ignores some nodes - e.g those with a higher total path-cost). I believe this is exactly what any correct implementation of A* would do, as it stops searching/expanding nodes as soon as the goal node is at the front of the priority queue (Not necessarily when it is first discovered - e.g when it is possible to achieve a lower path-cost via another node). Any nodes still in the queue or that have not been expanded yet can be safely ignored, as there cannot exist any route with a lower total path-cost (given that all intermediate path-costs are positive) as soon as the goal node is at the front of the queue.

II. Sort

InsertionSort: O(nk), where n is the number of elements to insert into the list (eg. children into queue), and k is the position the child is to be inserted into.

Binary: O(k log n) eliminate half of the list at each step. n is the number of items in the sorted list, and k is the number of items to insert.

Example: Sort 1-10 elements into a list with 1024 items. Correct position is 1;10; 11; 100.

	InsertionSort				Binary
1	1	10		10	100
10	10	100		10	100
11	11	110		10	100
100	100	1000		10	100

Quicksort: O(n log n) (I think)

III. Storage

Empty String: 40Bytes
Pointer/Reference: 32-64bit (depending on processor)

I'm guessing pointers are much better to pass around than Strings.

IV. Complexity

$O(log\, n)$:
$O(log\, 8) = 3$
$2^3 = 8$
$O(log\, 1024) = 10$
$1024 = 2^{10}$