CMSC351-2023-09-18

Monday, September 18, 2023 8:14 AM

Please try to be here so we can start at 9AM!

NOTES!

- Review Wednesday, Exam Friday!
- Today (Insertion Sort) is NOT on the exam!
- There is no HW due next Wednesday!
  (Insertion Sort will be on next week's HW)

INSERTION SORT

① Basic Idea Start w/ a list of length n:

A = [4, 3, 1, 8, 3, 2]

Start w/ A[1]=3. Pick it up. look left. if anything is >3, slide it right. Then insert 3 into the hole

A = [3, 4, 1, 8, 3, 2]
↑ insert!

Next A[2]=1. Pick up 1. Slide 3,4 to right. Insert 1.

A = [1, 3, 4, 8, 3, 2]
↑ insert!

Next A[3]=8. Pick up 8. Slide nothing! Insert 8.

A = [1, 3, 4, 8, 3, 2]
ins.

Next A[4]=3. Etc.

A = [1, 3, 3, 4, 8, 2]
ins.

Last A[5]=2 ETC

A = [1, 2, 3, 3, 4, 8]
ins.

DONE! :)

② Pseudocode

```
\\PRE: A is a list of length n.
for i = 1 to n-1                          n iterations
    key = A[i]                            } c1
    j = i-1                               }
    while j >= 0 and key < A[j]           ??? iterations at c2 each
        A[j+1] = A[j]                     } c3
        j = j - 1                         }
    end
    A[j+1] = key                          c4
end
\\POST: A is sorted.
```

Pick up A[i]

look to the left!

while we've not fallen off the front of the list and A[j] > key shift it right.

when while loop fails, j = index for the value which should **not** be moved.
we insert key to the right of this!

③ Time Complexity

Under blc ... how many times does the while loop iterate?!

BEST-CASE: How fast could this run?
while check fails immediately for each i.
This happens when the list is already sorted!
Then:

$$T(n) = (n-1)(c_1 + c_2 + c_4) = \Theta(n)$$

c_1: before while; c_2: check + fail; c_4: after while

WORST-CASE: How slow could this run?
while check passes until $j=-1$ for each i.
This happens when the list is reverse sorted!
Then:

$$T(n) = \sum_{i=1}^{n-1} [c_1 + (i)(c_2+c_3) + c_2 + c_4]$$

c_1: before while; $(i)(c_2+c_3)$: while loop passes for $j = i-1, i-2, \ldots, 0$, this is i times at c_2+c_3 each; c_2: while loop fails for $j=-1$, time c_2 to check; c_4: after while

$$= \ldots = \Theta(n^2)$$

AVG CASE: What is an "avg" list?!

First, for a list A, an inversion is a pair of elts which is out of sorted order.

e.g. [4,3,1,8,3,2] Inversions, given by indices (i,j) w/ $i<j$

4 out of order w/ 3,1,3,2. Index pairs $(0,1),(0,2),(0,4),(0,5)$

$(1,2),(1,5)$

Obs: 9 inversions. $(3,4),(3,5)$

$(4,5)$

next: if a list is sorted, #Inv = 0

if a list is reverse sorted, # Inv = $C(n,2) = \frac{n(n-1)}{2}$

(all pairs (i,j) w/ $i<j$)

perhaps: we can say an average list has $\frac{1}{2}C(n,2) = \frac{n(n-1)}{4}$ inv.!

Great, but how long does it take to run?!

We'll do one better! Let I = # inversions.

key point: each elt shift in the code corrects one inversion.

so: each time while loop passes, an inv. is fixed!

So:

$$T(n): \underbrace{(n-1)(c_1+c_4)}_{\text{alg but not the while loop}} + \underbrace{I(c_2+c_3)}_{\substack{\text{while loop passes} \\ I \text{ times, so} \\ c_2+c_3 \text{ to check} \\ + \text{execute body}}} + \underbrace{(n-1)c_2}_{\substack{\text{while loop fails} \\ \text{once for each } i \\ \text{so } n-1 \text{ times.}}}$$

So for our "avg" list:

$$T(n) = (n-1)(c_1+c_4) + \frac{n(n-1)}{4}(c_2+c_3) + (n-1)c_2$$

$$= \ldots = \Theta(n^2)$$

④ Misc

(A) Stable? YES!

(B) Aux Space? $\Theta(1)$

(C) In-Place? YES!