Lecture 3: Friday Jan 10, 2003

today

- insertion sort
  - WC/AC/BC run time
  - correctness
- next: mergesort
- next: asymptotic analysis

announcements

- seminars start Monday January 13
- quizzes start Monday January 20
- attend the seminar in which you are registered

insertion sort run time (unit cost RAM)

- key comparison (KC): `i>0 and A[i]>key`
- RAM run time prop'l to number KC

insertion sort best case (BC) KC

- one time for each j, so $n-1$

insertion sort worst case (WC) KC

- j times for fixed j, so $\sum_{j=2}^{n} j \; = \; n(n+1)/2-1$

insertion sort (AC) KC

- average case: always ask **"average over what distribution of inputs?"**
- unless stated otherwise, assume each possible input equiprobable
- here, each of $n!$ possible inputs equiprobable
- key observation: equiprobable inputs implies **for each key, rank among keys so far is equiprobable**
- e.g. $j=4$, exp. num. KC is $(1+2+3+4)/4=2.5$
- expected number of KC to insert key j is $(\sum_{r=1}^{j} r)/j \; = \; (j+1)/2$
- total expected number of KC is $\sum_{j=2}^{n}(j+1)/2 \; =$

  $(\sum_{j=2}^{n} j)/2 + (\sum_{j=2}^{n} 1)/2 \; = \; n(n+1)/4-1/2+(n-1)/2 \; = \; n(n+3)/4-1$

algorithm correctness

- always a good idea to verify correctness
- becoming more common in industry
- this course: a simple intro to correctness proofs
- when loop in involved, use loop invariant (and induction)
- when recursion involved, use induction

loop invariant (LI)

- initialization: does LI hold 1st time through?
- maintenance: if LI holds one time, does LI hold the next?
- termination #1: LI upon completion implies corrrectness?
- termination #2: does loop terminate?

insertion sort loop invariant

- at start of line 1, keys initially in `A[1...j-1]` are in `A[1...j-1]` and sorted

initialization

- `A[1...1]` is trivially sorted ☺

maintenance

- informally: body of outer loop works by moving A[j-1] A[j-2] ...one position to the right, until the proper position for A[j] is found
- exercise: give a more formal proof

term'n #1

- upon completion, execution reached line 1 with j=n+1
- so invariant holds for j=n+1, so all keys are sorted ☺

term'n #2

- for loop counter never altered, so loop terminates ☺

sketch of more formal proof of maintenance condition

- assume LI holds when for $j = k$ so A[1] $\leq$ A[2] $\leq \ldots \leq$ A[k-1]
- how to prove LI holds after next loop body execution?
- loop body contains another loop: use another LI!
- exercise: find a useful LI2
- exercise: prove LI2
- exercise: using LI2, prove LI1
- hint: when LI2 terminates, i=0 or A[i] $\leq$ key (and i=j-1 or A[i+1]>key)