Average Case Analysis

Two Examples

Sequential Search

Sequential Search Algorithm:

// searches for a given value in a given array by sequential search

// Input: An array A[0 ..*n*-1] and search key *K*

// Output: the index of the first element of *A* that matches *K*. or –1 if there are no matching elements

$i \leftarrow 0$

while ($i < n$) and ($A[i]$!= K) *do*

    $i \leftarrow i + 1$

if ($i < n$) return i

else return -1

Sequential Search

- Worst case: # of comparisons ???
- Best Case: # of comparisons???
- Average Case:
- Probability of successful search is p, $0 \le p \le 1$
- The probability of the first match occurring in the i^{th} position is the same for every i
- In the case of successful search, the probability of the first match occurring at the i^{th} position is p/n for all i, $0 \le i \le n - 1$

Sequential Search

- In the case of an unsuccessful search, the probability is $(1 – p)$ and the # of comparisons is n
- So $C_{avg}(n) = (1 * p/n + 2 * p/n + \dots n * p/n) + n * (1 – p)$

$$= p/n * (n * (n+1))/2 + n * (1 – p)$$

$$= \frac{p * (n+1)}{2} + n * (1 – p)$$

Sequential Search

- If p = 1, # of comparisons is (n+1)/2 and if p = 0, then # of comparisons is n.
- Realize that average case efficiency cannot be determined by taking the average of best and worst cases.
- Usually it is much better than the pessimistic worst case efficiency.

Insertion Sort

- For simplicity, assume that the array is indexed from 1 to *n*.
- Here is the pseudo code:

for i = 2 to n do

    $key = a[i]$

    insert *key* at the appropriate place in $a_1,\ldots a_i$

Insertion Sort

for (j = 2; j <= n; j++) {

key = $a[j]$;

$i = j - 1$;

while (($i > 0$) && ($a[i] > key$)) **do** {

$a[i+1] = a[i]$;

$i = i - 1$;

}

a[i+1] = *key;*

To find the average number of comparisons made by insertion sort we need to ask: **how many comparisons it takes to move the j^{th} element into position?**

At most:????? At least: ?????

Insertion sort

- How many possible positions are there for the j^{th} element?
- For $j = 2$: 2 possibilities: 1 or 2
- For $j = 3$: 3 possibilities: 1, 2 or 3
- …
- For $j = i$: i possibilities: 1,2,… i
- How many comparisons does it take to get to each of these j possible positions?

Insertion Sort

- For the j^{th} element, the algorithm will make I, 2, … (j -1) comparisons for locations j, (j – 1), …2 and it will do (j – 1) comparisons for the 1st position (note that every one of these possibilities has equal probability)
- Therefore, the average # of comparisons to insert the j^{th} element:
- $= 1/j \sum i \quad + 1/j \ * (j - 1) \qquad 1 \leq i \leq j\text{-}1$
- $= (j - 1)/2 + 1 - 1/j$
- This needs to be summed for $j = 2$ to n

Insertion Sort

- # of comparisons on average:
- $O(n^2)$