Parma University, Math. Deptartment

Algorithms and Data Structures (Binary search and Merge Sort)

Marco Pellegrini

Summary of lecture 4

- Slow searching: sequential search
  Fast searching: Binary Search,
  Fast Sorting: Merge-Sort,
  Analysis of Merge-Sort,
  Dictionary Abstract Data Structure.

Why is sorting important?

- Sorting makes searching much, much faster.
- Example: an array A with 8 numbers, unsorted:
  A = [15, 10, 2, 9, 7, 33, 77, 12].
- To check if a number x is in A we have to check in the worst case all 8 positions, for example by SEQUENTIAL SEARCH.
- When A is sorted:
  A = [2, 7, 9, 10, 12, 15, 33, 77],
- To know if a number x is in A we have to check in the worst case 4 positions by using BINARY SEARCH.

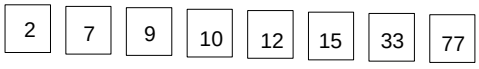
Sequential search and binary search

```
SEQUENTIAL-SEARCH(A: array, x: number)
  FOR i := 0 TO Length(A) -1
    DO IF A[i] = x THEN RETURN True
  END FOR
  RETURN False
```

```
BINARY-SEARCH(A: array, x: number)
  i := 0
  j := length(A)-1
  WHILE i < j
    DO k := ⌊ (i+j)/2 ⌋
       IF A[k] = x THEN RETURN True
       IF A[k] > x THEN j :=k-1
       IF A[k] < x THEN i :=k+1
  END WHILE
  IF A[i] = x THEN RETURN True
          ELSE RETURN False
```


Binary Search

an example



Look for numbers: 22, 10, 3, 24
using sequential search and binary search

How many boxes did you check?

Sequential Search

Analysis

- Sequential search. If x is not in the array A we need to check all elements in A to make sure that x is not in A.
  Sequential Search is WORST-CASE linear.
- If x is in A then on average we visit (n+1)/2 entries of A to find it. So:
  $T_{AVERAGE}(n) = (n/2)\text{Prob}(x \text{ in } A) +$
  $n\ \text{Prob}(x \text{ not in } A)$.
- Even in the average case sequential search is Linear

Binary Search

analysis

- To analyze BINARY-SEARCH we consider the quantity D=(j-i).
- At time 0, before WHILE LOOP D(0)= n-1 < n.
- Every time the WHILE LOOP is executed D is at least halved: $D(t+1) \leq D(t)/2$.
- We exit the WHILE LOOP when $D(T) \leq 1$ for some value T.
- Solving $n/2^T \leq 1$. We get T = log n.
- BINARY-SEARCH is a LOGARITHMIC procedure both in WORST CASE and on AVERAGE.

Merge-Sort

- We saw in lecture 1 INSERTION-SORT:
  1) Sorts in place (i.e uses only 1 array)
  2) Is quadratic in WORST-CASE
  3) Is quadratic in AVERAGE-CASE
  4) It has two nested for-loops, so counting instructions was easy.
- Today we will see how to sort numbers by using MERGE-SORT.
- We will see the Divide-and-Conquer paradigm.
- We will see how to analyze RECURSIVE procedures by using RECURSIVE equations.
- We will compare MERGE-SORT and INSERTION-SORT.

Merge Sort

- The basic idea is that to MERGE two arrays that are already sorted is easy.
-

```
MERGE(A,B: sorted arrays): sorted array
  i :=0, j := 0, k :=0
  a := length(A)- 1, b:= length(B) -1,
  WHILE i ≤  a AND j ≤ b
    DO  IF A[i] < B[j]
        THEN C[k] := A[i], i:=i+1, k:=k+1
        ELSE C[k] := B[j], j:=j+1, k:=k+1
  END WHILE
  IF i > a THEN COPY B[j,..,b] INTO C[k...]
  IF j > b THEN COPY A[i,...a] INTO C[k...]
  RETURN C
```


Merge-Sort

- If an array has only 1 element is already sorted.

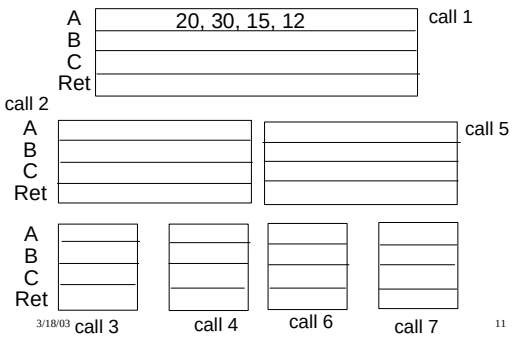
```
MERGE-SORT(A: array of numbers): sorted array
 a := length(A)
 IF a=1 THEN RETURN A
 q := ⌊a/2⌋

 B = MERGE-SORT(A[0,..,q-1])
 C = MERGE-SORT(A[q,..a-1])
 RETURN MERGE(B,C)
```

- We assume that our language allows to define recursive functions (i.e. functions that can call themselves in their definition).

Exercise: simulate the calls of merge-sort

Fill in the data

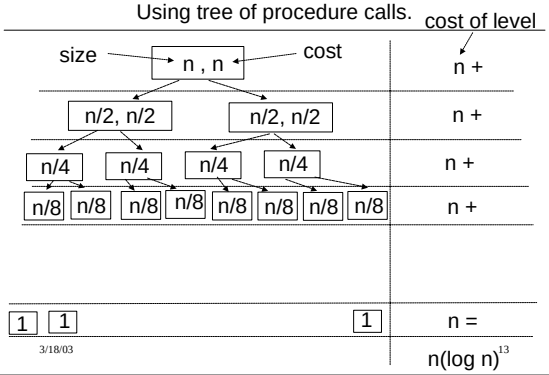


Time analysis of MERGE-SORT

- Procedure MERGE takes LINEAR Time. Why?
- Procedure MERGE-SORT takes time O(1) if n=1. Otherwise it call itself two times on arrays that are half of the initial one. Then it calls MERGE that takes linear time.
- Let us call T(n) the time needed to run MERGE-SORT. We have:
- T(1) = O(1).
- T(n) = O(1) + 2T(n/2) + O(n), if n>1.
- We have written a RECURSIVE EQUATION for T(n). The problem now is to solve the equation

Solution of T(n) = 2T(n/2) + n

Using tree of procedure calls.



Exercise

- Use the tree of procedure calls to solve the recurrence $T(n) = 2T(n/2) + n^2$, where $T(1) = 1$.

Observations on MERGE-SORT

The time bound of Merge-Sort is in the *best case, worst case* and *average case* O(n log n).

We can conclude that Merge-Sort is asymptotically more efficient than Insertion-Sort.

Merge-Sort is an example of a general algorithmic technique: Divide and Conquer.

DIVIDE the problem into a certain number of subparts.

CONQUER the subproblem by recursive calls of the procedure.

COMBINE the partial solutions into the global one.

Observations on three different analysis styles

- We have analyzed INSERTION-SORT by direct counting of the number of operations (number of executions of FOR loops.
- We have analyzed BINARY-SEARCH by analyzing a function D, decreasing with the execution of the WHILE loop.
- We have analyzed MERGE-SORT by using recursive equations and one method to solve the recurrence (tree of procedure calls).

A New Abstract Data Type: Dictionary

- A dictionary is an ADT supporting the following operations:
  1) MEMBER(x,D) returns true if x is in D.
  2) PREDECESSOR(x,D) finds the smaller y in D which is greater than x.
  3) SUCCESSOR(x,D) finds the larger y in D which is smaller than x
  4) CREATE(N,D) given the set of numbers N create its dictionary.
- We can use ARRAY, SORTING and BINARY-SEARCH as the corresponding DS.
  We can build the data structure Sorted Array in time O(n log n). We can implement MEMBER, PREDECESSOR, SUCCESSOR using BINARY-SEARCH in time O(log n).

A Second Data Structure

Priority Queue

- A Priory Queue is an organization of a data set S to perform the following operations:
- INSERT(x,S): insert x in the data structure S.
- MINIMUM(S): return the smallest value in S.
- DELETE-MIN(S): delete the smallest value in S.
- MAKE-EMPTY(S): build an empty priority queue.
- Applications: Sorting (HEAP-SORT).
  Data Compression (Huffman codes).
  Also in Graph algorithms, Computational Geometry.
- Next lecture: the HEAP implementation of P-Queues.

Conclusions

- Sorted arrays allow a fast type of search called BINARY-SEARCH.
- We have seen the MERGE-SORT procedure that takes time O(n log n) but does not sort in place.
- MERGE-SORT is an example of Divide and Conquer.
- MERGE-SORT is recursive and is analyzed using recursive equations.
- Static Dictionary and Priority Queue

HW1.3 (1.3-7 CLR)

- You are given an array A of n numbers and a number x. Determine if there are two numbers in A whose sum is x.
- The running time of your method should be O(n log n).
  [Hint: use sorting and binary-search]