Divide and Conquer Pattern applied to Sorting Equals the MergeSort Algorithm

Project 3 Management and Data Structures

CSI 310: Lecture 12 (Spr 05)
Only true "logically"... this data is actually stored in the called activation.

ACTIVATION is destroyed, and its Record gets recycled.

Really: When an ACTIVATION executes the return; operation, that
spot within the function's body if this activation CALLED a function
DIFERENT ONE) whose CALL operation created this one. (3) The return
activation's local (i.e., automatic) variables. (2) A pointer to the activation (A
An Activation Record: The data structure that holds (1) An

Definition of Activation Record: The data structure that holds (1) An

Wrong: Control "jumps" or "goes to" the function's body.

What HAPPENS when the computer executes a FUNCTION CALLED?
3. A stack is a sequence for which insertion and deletion are only done at ONE

ENd (called the top).

2. Activation records are stored in a STACK (like a pile of bills).

1. Current Lab (4): Observe the activation records for runs of the recursive

function write-verbatim() from DSO chapter 9.
5. Code seetionsort in with pre and post conditions. Implementation.

The print operation lets you and the TA TEST your input phase

arcs, char *arcs[]

command line argument array (pointed to by 2nd arg of main) in the

command phase: dispatches the operations listed ALREADY in the

THE capability

capacity when necessary as explained in ch. 4 of DS0. Great idea: MAKE

3. The SortBench dynamic partially filled array MUST increase its own

SortBench::append(char)

2. Input line phase. Line reader and sorter. (Sorter calls

of char pointers (or string pointers).

One object of class SortBench will have one dynamically partially filled array

SortBench.

Create and use header file SortBench.h to declare the class

Project 3 Management

University at Albany Computer Science Dept.
May be write a test driver for merge.

8. Implementation, test, and debugging of FIRST and SECOND merge sort.

7. Write interfaces and pre/post conditions for the merge sort() and merge() functions of Ch. 13, except the apply to arrays of char * or string * not to arrays of int.

6. Follow instructions to implement sort algorithms timing.

\[
\text{void selectsortsort (string data[], int n)} \text{;}
\]

\[
\text{void selectsortsort (char data[], int n)} \text{ as explained in DS0 13.1}
\]
start

exit

command phase

input phase

read and process text

mergesort not here.

Note1: mergesort is in progress.

Note2: 2 dynamic *char[] used by mergesort() are shown.

Note3: argc, argv, etc. on Handout, not here.

sorterbench

mergesort

selectionsort

print

print

loop thru and do commands

from argv[]

command phase

sorterbench

datap

used

capacify

6

3

capacity

Lines

Process Text

Read and

input phase

Lines

State diagram

Data structure diagram

Note1: mergesort is in progress.

Note2: 2 dynamic *char[] used by mergesort() are shown.

Note3: argc, argv, etc. on Handout, not here.
{ return
    count = // A sorted string was printed:
    Now, A[0].nch-1 IS SORTED [1].nch-1
} Now, A[1] has the smallest char from A[1].nch-1
*/ int i = 1; // code to "swap" A[minspos], A[i] */
{
    int minspos = i;
    if (A[i] < A[minspos])
        for (int j = i+1; j < nch; j++)
            if (A[j] < A[minspos])
                minspos = j;
    if (i < nch-1)
        for (int i = 0; i < nch-1; i++)
            if (nch = strlen(A));
    return i; int nch = strlen(A);
}
const int nsize = 100; char A[nsize]

main()
    using namespace std;
    #include <cstdlib>
    #include <iostream>
    #include <cstring>

    //Selection sort demo: processes characters within the array A.
We will then illustrate the (recursive) MergeSort half.

We now illustrate what half your Project 3 work must do.

It uses the Selection Sort algorithm for sorting.

This program manipulates characters as if they were numbers.
Sample list of items to sort:

cat
armadillo
bat
ape

caterpillar
dog
ant
zebra
Many of fixed capacity for temporary storage of sequences.

(a) One of dynamic capacity for inputting an unpredictable number of
strings.

3. Dynamic arrays for

2. The fastest way to change the positions of STRINGs sequenced with an
array is to store POINTERS (Java references) in the array, and move
pointers/re value.

1. Implement the ADT sequence (element collection where ORDER
Matters and multiple occurrences are OK) with an array.

Main data structure ideas for sorting strings using an array:

(p) Many of fixed capacity for temporary storage of sequences.

University at Albany Computer Science Dept.
is called "Merging"

3. Combine the two sorted groups into one large sorted list. 

INDEPENDENTLY!

2. Sort each of these smaller groups (by recursive calls). That means size.

I. Divide the elements to be sorted into two groups of equal (or almost equal) 

paradigm or pattern is applied to the problem of sorting a sequence: 

Merge sort is the algorithm that is invented when the divide-and-conquer
exception bad_alloc is thrown.  

NOTE: If there is insufficient dynamic memory, then

elements REARANGED in SORTED order. 

POST: data[0 ⋯ (n1+n2-1)] contain the original

data[n1 ⋯ (ni + nz -1)] are also ALREADY SORTED. 

data[n1 ⋯ (ni-1)] are ALREADY SORTED!!! and 

(CRITICAL!!!)

PRE: data points to subarray with at least (ni + nz) elements.

void merge(int data[], size-t n1, size-t n2)

DSO's example:
exception bad_alloc is thrown;

NOTE: If there is insufficient dynamic memory, then

[ data[0] => ... => data[n-1] ]

// Original elements rearranged in sorted order. I.E.
// Post: The elements of data[0] ... (n-1) contain its
// Pre: data==adder. of an array of length at least n.

void mergesort(int data[], size_t n);