Merge sort (C)

From LiterateProograms

> **Other implementations**: ACL2 | **C** | C++ | dc | Eiffel | Erlang | Haskell | Java | JavaScript | Lisp | OCaml | Oz | Perl | Prolog | Python | Ruby | Scheme

Merge sort or mergesort is a simple but efficient sort algorithm that splits the list into two sublists, sorts each one, then combines them into a single sorted list. It has good worst-case performance and requires only sequential access, making it ideal for sequential data structures like linked lists. In this C example we'll write a simple mergesort for arrays.

In these examples we will demonstrate the sort only for *int*s; in the future this may be generalised to arbitrary data.

To sort an array using mergesort, we split the array into two subarrays in an implicit way:

```
<<simple array mergesort>>=
void mergesort_array(int a[], int size) {
    check termination condition
    mergesort_array(a, size/2);
    mergesort_array(a + size/2, size - size/2);
    merge the two sorted sublists
}
```

The tricky part is merging the two sorted subarrays. We have two ways to go about this:

1. Create a separate buffer the same size as the list for constructing the result of the merge in.
2. Attempt to rearrange elements as we go so that the portion of the merged list constructed so far does not overwrite sublist elements that are not yet merged.

The first option is simple but has a high memory overhead (linear auxilary memory) as well as a problem with frequent allocations. The second option is in-place and more efficient, but also more complicated. Here we stick to the simpler method, but deal with the overhead of allocations by allocating a single buffer the size of the original list which is reused by every call:

```
<<array mergesort>>=
void mergesort_array(int a[], int size, int temp[]) {
    mergesort_array variable declarations
    check termination condition
    mergesort_array(a, size/2, temp);
    mergesort_array(a + size/2, size - size/2, temp);
    merge the two sorted sublists into temp
    memcpy(a, temp, size*sizeof(int));
}
```

The `memcpy` at the end puts the final result back in *a*, and requires a header:

```
<<header files>>=
#include <string.h>
```

The merge operation itself is straightforward. We have two pointers starting at the beginning of each subarray, and we continually copy the smaller value into the result array and advance that value's pointer:

```
<<mergesort_array variable declarations>>=
int i1, i2, tempi;
<<merge the two sorted sublists into temp>>=
i1 = 0;
i2 = size/2;
tempi = 0;
while (i1 < size/2 && i2 < size) {
    if (a[i1] <= a[i2]) {
        temp[tempi] = a[i1];
        i1++;
    } else {
        temp[tempi] = a[i2];
        i2++;
    }
    tempi++;
}
```

If either pointer hits the end of its subarray before the other, the remaining values are simply copied from the remaining subarray:

```
<<merge the two sorted sublists into temp>>=
while (i1 < size/2) {
    temp[tempi] = a[i1];
    i1++;
    tempi++;
}
while (i2 < size) {
    temp[tempi] = a[i2];
    i2++;
    tempi++;
}
```

As with other divide-and-conquer sorts, the simplest termination condition is when the list becomes of size one or less:

```
<<simple check termination condition>>=
if (size <= 1) {
    return;
}
```

But more efficient is to switch to insertion sort for small enough lists, like this code from Insertion sort (C, simple):

```
<<constants>>=
#define MIN_MERGESORT_LIST_SIZE    32

<<check termination condition>>=
if (size < MIN_MERGESORT_LIST_SIZE) {
    /* Use insertion sort */
    int i;
    for (i=0; i < size; i++) {
        int j, v = a[i];
        for (j = i - 1; j >= 0; j--) {
           if (a[j] <= v) break;
            a[j + 1] = a[j];
```

```
        }
        a[j + 1] = v;
    }
    return;
}
```

This completes the sort. We can try it out with this sample code, which generates a random array of a size specified on the command line, sorts it, then verifies that it is in order:

<<mergesort_array_test.c>>=

header files

```
#include <time.h>    /* time() */
#include <stdlib.h>  /* rand() */
#include <stdio.h>   /* puts() */
```

constants

array mergesort

```
int main(int argc, char* argv[]) {
    int size = atoi(argv[1]);
    int* a    = malloc(sizeof(int)*size);
    int* temp = malloc(sizeof(int)*size);
    int i;
    srand(time(NULL));
    for(i=0; i<size; i++) {
        a[i] = rand() % size;
    }
    mergesort_array(a, size, temp);
    for(i=1; i<size; i++) {
        if (!(a[i-1] <= a[i])) {
            puts("ERROR");
            return -1;
        }
    }
    puts("SUCCESS");
    return 0;
}
```

Download code (http://en.literateprograms.org/index.php?title=Special:Downloadcode/Merge_sort_(C)&oldid=17235)

Retrieved from "http://en.literateprograms.org/index.php?title=Merge_sort_(C)&oldid=17235"
Categories: Merge sort | Programming language:C