Radix sort (C)

From LiterateProograms

Other implementations: **C** | Java

An implementation of radix sort (http://en.wikipedia.org/wiki/Radix_sort) for unsigned int in c. It uses counting sort (http://en.wikipedia.org/wiki/Counting_sort) for each iteration (see Category:Counting sort for more implementations of counting sort).

<<radixsort.c>>=

```
#include<stdlib.h>
#include<stdio.h>
#include<assert.h>
#include<limits.h>
#include<strings.h>
#include<string.h>
```

radix_sort_uint

test

Contents

radix_sort_uint()

This function takes the pointer to, and size, of an array, and the number of bits used as the key in each iteration.

<<radix_sort_uint>>=

```
void radix_sort_uint(unsigned int *a, size_t size, int bits)
{
        unsigned int mask;
        unsigned int rshift=0u;
        unsigned int *p, *b, *b_orig;
        unsigned int i;
        unsigned int key;
        int cntsize;
        int *cntarray;
```

Memory

We use an array of the same size as the original array to store the result of each iteration.

```
<<radix_sort_uint>>=
        b=b_orig=malloc(size*sizeof(unsigned int));
```

An array is needed to store the count for each key value.

```
<<radix_sort_uint>>=
        cntsize=1u<<bits;
        cntarray=calloc(cntsize, sizeof(int));

        assert(cntarray);
        assert(b_orig);
```

The main algorithm

mask is the bitmask used to extract the sort key. We start with the *bits* least significant bits and left-shift it the same amount at each iteration. When all the bits are shifted out of the word, we are done.

```
<<radix_sort_uint>>=

        for(mask=~(UINT_MAX<<bits); mask; mask<<=bits, rshift+=bits) {
```

Counting sort

We count each key value.

```
<<radix_sort_uint>>=
                for(p=a; p<a+size; ++p) {
                        key=(*p & mask)>>rshift;
                        ++cntarray[key];
                }
```

Here, we sum up how many elements there are with lower key values, for each key.

```
<<radix_sort_uint>>=

                for(i=1; i<cntsize; ++i) cntarray[i]+=cntarray[i-1];
```

The values in *cntarray* are used as indexes for storing the values in *b*. *b* will then be completely sorted on this iteration's key. Elements with the same key value are stored in their original internal order.

```
<<radix_sort_uint>>=

                for(p=a+size-1; p>=a; --p) {
                        key=(*p & mask)>>rshift;
                        b[cntarray[key]-1]=*p;
                        --cntarray[key];
                }
```

Cleanup

We swap the *a* and *b* pointers, so that the next iteration works on the current *b*, which is now partially sorted.

<<radix_sort_uint>>=

```
                p=b; b=a; a=p;
```

cntarray is cleaned up for the next iteration.

<<radix_sort_uint>>=

```
                bzero(cntarray, cntsize * sizeof(int));
        }
```

If the completely sorted array is in the malloc'ed array, we must copy it to the array provided by the user.

<<radix_sort_uint>>=

```
        if(a==b_orig) memcpy(b, a, size*sizeof(unsigned int));

        free(b_orig);
        free(cntarray);
}
```

Test driver

This test program uses 4 bits for the key. *radix_sort_uint()* will accept any number of bits for the key, but it will allocate at least 2^{bits} *int*s of storage.

<<test>>=

```
int main()
{
        int i;
        unsigned int a[]={
                123,432,654,3123,654,2123,543,131,653,123,
                533,1141,532,213,2241,824,1124,42,134,411,
                491,341,1234,527,388,245,1992,654,243,987};

        printf("Before radix sort:\n");
        for(i=0; i<sizeof a/sizeof(unsigned int); ++i)
                printf(" %d", a[i]);
        putchar('\n');

        radix_sort_uint(a, sizeof a/sizeof(int), 4);

        printf("After radix sort:\n");
        for(i=0; i<sizeof a/sizeof(unsigned int); ++i)
                printf(" %d", a[i]);
        putchar('\n');

        return 0;
}
```

Download code (http://en.literateprograms.org/index.php?title=Special:Downloadcode/Radix_sort_(C)&oldid=4760)

Retrieved from "http://en.literateprograms.org/index.php?title=Radix_sort_(C)&oldid=4760"
Categories: Programming language:C | Environment:Portable | Radix sort

- This page was last modified on 15 May 2006, at 02:59.
- Content is available under the MIT/X11 License.