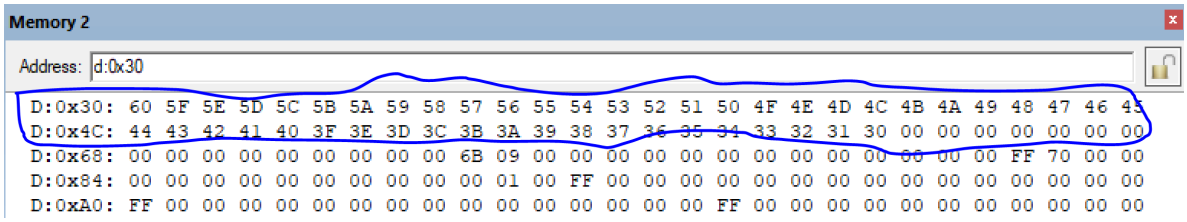
源代码在最后。

首先，对于30H~60H区域，进行赋值，由于排完序后数据应该是升序，因此这里特意以降序形式进行赋值。

```
for(i=0x60;i>=0x30;i--){
        *sadd=i;
        sadd++;
    }//decreasing order,here I assign a value to the memory unit in reverse order.
```

赋值后的存储空间如下：




接下来进行排序：

我采用了两种排序算法，

冒泡排序和快速排序：

冒泡排序：

```
for(i=0;i<0x31;i++)
        for(k=0;k<0x31-i-1;k++){
            if(add[k]>add[k+1]){
                temp=add[k+1];
                add[k+1] = add[k];
                add[k] = temp;
            }
        }
```

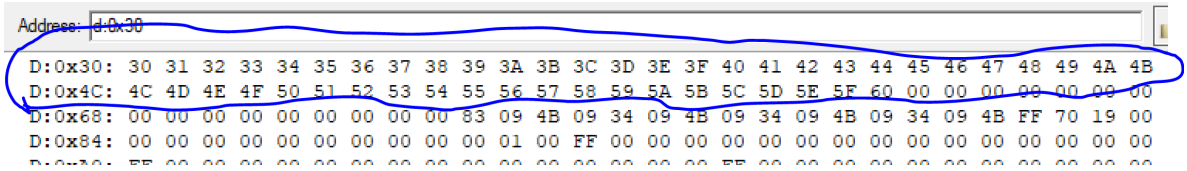
快速排序：

```
void quick_sort(unsigned char* a,unsigned char left,unsigned char right)
{
    unsigned char index;
    if(left >= right)
    {
        return;
    }
    index = getindex(a,left,right);//get the key'index
    if(index>left)//Judge whether it is out of bounds
    {
    quick_sort(a,left,index - 1);}
    if(index<right){
    quick_sort(a,index + 1,right);//Recursive
    }
}
```

```c
unsigned char getindex(unsigned char* a,unsigned char left,unsigned char right)
{
    unsigned char key = a[right];
    unsigned char r = right;
    unsigned char temp;

    while(left < right)
    {
        while(left < right && a[left] <= key)
        {
            ++left;
        }
        while(left < right && a[right] >= key)
        {
            --right;
        }
        temp  = a[right];
        a[right] = a[left];
        a[left] = temp;
    }
    temp  = key;
    a[r] = a[left];
    a[left] = temp;
    return left;
}
```

排序后效果：




完成排序任务。

问题1采用什么算法完成排序最快？

采用快速排序，堆排序，归并排序最快，因为时间复杂度为o(NlogN)

问题2采用什么算法最合适？

采用堆排序最合适，因为空间复杂度为o(1)，而快速排序的空间复杂度为o（NlogN）,归并排序的空间复杂度为o(N)

源代码：

```c
#include<reg51.h>
data unsigned char ADDR _at_ 0x30;
//begin with 0x30
unsigned char getindex(unsigned char* a,unsigned char left,unsigned char right)
{
    unsigned char key = a[right];
    unsigned char r = right;
    unsigned char temp;

    while(left < right)
    {
        while(left < right && a[left] <= key)
        {
```

```
            ++left;
        }
        while(left < right && a[right] >= key)
        {
            --right;
        }
        temp  = a[right];
        a[right] = a[left];
        a[left] = temp;
    }
    temp  = key;
    a[r] = a[left];
    a[left] = temp;
    return left;
}
void quick_sort(unsigned char* a,unsigned char left,unsigned char right)
{
    unsigned char index;
    if(left >= right)
    {
        return;
    }
    index = getindex(a,left,right);//get the key'index
    if(index>left)//Judge whether it is out of bounds
    {
    quick_sort(a,left,index - 1);}
    if(index<right){
    quick_sort(a,index + 1,right);//Recursive
    }
}
void main(){
    unsigned char i;
    unsigned char k;
    unsigned char temp;
    unsigned char *sadd;
    unsigned char *add;
    SP=0x70;
    sadd =&ADDR;
    add=sadd;//0x30
    for(i=0x60;i>=0x30;i--){
        *sadd=i;
        sadd++;
    }//decreasing order,here I assign a value to the memory unit in reverse
order.

    //bubble sort
    /*for(i=0;i<0x31;i++)
        for(k=0;k<0x31-i-1;k++){
            if(add[k]>add[k+1]){
                temp=add[k+1];
                add[k+1] = add[k];
                add[k] = temp;
            }
        }*/

    //quick sort
    quick_sort(add,0,0x30);
        while(1);
```

```
    //we succeed in sorting!
}
```