[**快速排序里的学问：随机化快排**](http://www.nowamagic.net/librarys/veda/detail/2398)

前一篇文章讲到了选择枢纽元的几种方法，其实第二种是随机选择元素作为枢纽元。那么在这篇文章里就实现一个[随机化排序](http://www.nowamagic.net/librarys/veda/tag/%E9%9A%8F%E6%9C%BA%E5%8C%96%E6%8E%92%E5%BA%8F)。

算法与前面《算法导论》里的例子差不多，只是在调用分割Partition时加入一个随机数，具体可以参看程序。

C语言代码为：

#include "stdio.h"

#include "math.h"

#include "stdlib.h"

int num = 10;

void swap(int \*a,int \*b)

{

int tmp;

tmp = \*a;

\*a = \*b;

\*b = tmp;

}

void PrintArray(int arr[])

{

int i;

for(i=0; i < num; ++i)

{

printf("%d ", arr[i]);

}

}

int Partition(int \*arr, int beg, int end)

{

int j;

int sentinel = arr[end];

int i = beg-1;

for(j=beg; j <= end-1; ++j)

{

if(arr[j] <= sentinel)

{

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i+1], &arr[end]);

printf("\n排序过程:");

PrintArray(arr);

return i+1;

}

int RandomPartition(int \*arr, int beg, int end)

{

int i = beg + rand() % (end-beg+1);

swap(&arr[i], &arr[end]);

return Partition(arr, beg, end);

}

void RandomQuickSort(int \*arr, int beg, int end)

{

if(beg < end)

{

int pivot = RandomPartition(arr, beg, end);

printf("\n随机选择 arr[%d](%d)", pivot, arr[pivot]);

RandomQuickSort(arr, beg, pivot-1);

printf("\n随机选择 arr[%d](%d)", pivot, arr[pivot]);

RandomQuickSort(arr, pivot+1, end);

}

}

int main()

{

int i;

int arr[10];

srand(time(0));

for(i=0; i < 10; i++)

{

arr[i] = rand()%100+1;

//printf("%d ", rand()%100+1);

}

printf("初始数组:");

PrintArray(arr);

RandomQuickSort(arr, 0, num-1);

printf("\n最后结果:");

PrintArray(arr);

return 0;

}

程序运行结果：

初始数组:79 36 68 39 10 96 59 60 84 21

排序过程:79 36 68 39 10 59 60 21 84 96

随机选择 arr[8](84)

排序过程:21 10 36 39 79 59 60 68 [84] 96

随机选择 arr[2](36)

排序过程:10 21 [36] 39 79 59 60 68 84 96

随机选择 arr[1](21)

随机选择 arr[1](21)

随机选择 arr[2](36)

排序过程:10 21 [36] 39 79 59 60 68 84 96

随机选择 arr[3](39)

随机选择 arr[3](39)

排序过程:10 21 36 [39] 68 59 60 79 84 96

随机选择 arr[7](79)

排序过程:10 21 36 39 60 59 68 [79] 84 96

随机选择 arr[6](68)

排序过程:10 21 36 39 59 60 [68] 79 84 96

随机选择 arr[4](59)

随机选择 arr[4](59)

随机选择 arr[6](68)

随机选择 arr[7](79)

随机选择 arr[8](84)

最后结果:10 21 36 39 59 60 68 79 [84] 96

Process returned 0 (0x0) execution time : 0.582 s

Press any key to continue.

一般来说随机选取[枢纽元](http://www.nowamagic.net/librarys/veda/tag/%E6%9E%A2%E7%BA%BD%E5%85%83)这种策略非常安全，除非随机数生成器有问题（这不像你所想象的那么罕见），因为随机的枢纽元不可能总在接连不断地产生劣质的分割。另一方面，随机数的生成一般是昂贵的，根本减少不了算法其余部分的平均运行时间。

比如上面程序的运行结果，可以看到，产生了不少随机数是对排序没有产生有效作用的，而产生这些随机数也耗费了不少时间。当然你也可以选择优化随机数生成器，这样又会引起更多的研究了。