1.给出不同策略实现两个整形量值的交换。

问题分析：

我们在日常生活中经常遇到两个变量交换其值的情况。但是我们用的高级语言不如基层语言汇编可以直接用XHGE来对两个值进行交换。我们必须采用一定的方法来实现。我在这里提出了三种交换方法。

方法一

方法一就是我们最常使用的用第三个变量做桥梁来实现。

```
int a = 5;
int b = 6;
int r ;
r = a ;
a = b;
b =r ;
cout<<a<<" "<<b;
```

方法二

方法二采用数学运算实现两个变量值的交换

```
int a = 5;
int b = 6;
a = a + b ;
b = a - b;
a = a - b;
cout<<a<<" "<<b<<endl;
```

方法三

位运算：也可以使用位运算来实现两个整数的交换

```
int a = 5;
int b = 6;
a = a ^ b;
b = a ^ b;
a = a ^ b;
cout<<a<<" "<<b<<endl;
```

2.给出一算法返回一元素序列中的两个最小值。

算法分析1：

我们经常遇到返回元素序列中一个最大（小）值的情况，这样的情况下我们只需要遍历比较，将最大（小）的元素放入其中即可。在本题中需要返回两个最小值，那么我们同样也可以设置两个变量分别保存最小，次小。在遍历的时候往里面存数即可。

算法描述1：

1.定义两个变量 Min1（最小） Min2（次小） 并赋予初值正无穷

2.遍历序列中的每个元素，如果该元素m小于Min1，则Min2 = Min1

Min1 = m 。如果该元素m大于Min1 小于Min2 ，则Min2 = m。如果该元素大于Min2，则不赋值。

3.遍历完成一遍后，饭后Min1和Min2的值。

因为只用遍历一次序列，所以时间复杂度：O（n）

代码实现：

```cpp
#include <iostream>
#include <climits>

using namespace std;

void find_two_min(int nums[], int n, int& min1, int& min2) {
    min1 = INT_MAX;
    min2 = INT_MAX;
    for (int i = 0; i < n; i++) {
        if (nums[i] < min1) {
            min2 = min1;
            min1 = nums[i];
        } else if (nums[i] < min2) {
            min2 = nums[i];
        }
    }
}

int main() {
    int nums[] = {3, 1, 5, 2, 4};
    int min1, min2;
    find_two_min(nums, 5, min1, min2);
    cout << "最小的两个数是" << min1 << " and " << min2 << endl;
    return 0;
}
```

算法分析二

分治算法：我们将数组分成两半，分别找到每一半的最小值和次小值，然后将这四个数进行比较，得出整个数组的最小值和次小值。其中一个常用的分治算法是归并排序，在归并排序中，最小值和次小值可以在合并两个已排序的子数组时进行查找。

算法描述二

如果序列长度为1，则直接返回最小值和次小值

如果序列长度为2，则将这两个数进行比较后返回

如果序列长度大于2，则我们采用递归的算法，将该序列均分为两个序列，再分别调用find_two_min函数，直到所分的序列长度为1或2

代码实现

```cpp
#include <iostream>
#include <climits> // 包含 INT_MAX 和 INT_MIN

using namespace std;

// 在指定的区间内，返回数组中的最小值和次小值
void find_two_min(int nums[], int l, int r, int& min1, int& min2) {
    // 如果指定区间中只有一个元素，则该元素既是最小值也是次小值
    if (l == r) {
        min1 = nums[l];
        min2 = nums[l];
        return;
    }

    // 如果指定区间中只有两个元素，则分别比较它们的大小，得到最小值和次小值
    if (l == r - 1) {
        min1 = min(nums[l], nums[r]);
        min2 = max(nums[l], nums[r]);
        return;
    }

    // 如果指定区间中有多个元素，则将其划分为两个区间进行递归求解
    int mid = (l + r) / 2;
    int left_min1, left_min2, right_min1, right_min2;
    find_two_min(nums, l, mid, left_min1, left_min2); // 递归处理左区间
    find_two_min(nums, mid + 1, r, right_min1, right_min2); // 递归处理右区间

    // 合并左右区间的结果，得到整个区间内的最小值和次小值
    if (left_min1 <= right_min1) {
        min1 = left_min1;
        min2 = min(left_min2, right_min1);
    }
    else {
        min1 = right_min1;
        min2 = min(right_min2, left_min1);
    }
}

int main() {
    int nums[] = { 3, 4, 5, 1, 2 }; // 待查找的数组
    int min1, min2; // 用来存储最小值和次小值
    find_two_min(nums, 0, 0, min1, min2); // 在整个数组中查找最小值和次小值
    cout << "最小的两个数为 " << min1 << " and " << min2 << endl;
    return 0;
}
```

这种算法的时间复杂度为O(n log n)