排序算法 8个

```
//直接插入排序
void InserSort(int R[], int n)
{
    int i, j;
    int temp;
    for(i = 1; i < n; ++i)
    {
        temp = R[i];
        j = i - 1;
        while(j >= 0 && temp < R[j])
        {
            R[j + 1] = R[j];
            --j;
        }
        R[j + 1] = temp; // 插入
    }
}

// 折半查找排序
void InsertSort(int A[]){
    int n = A.size();
    int low, high;
    for(int i = 2; i < n; ++i){
        A[0] = A[i];
        low = 1;
        high = i - 1;
        while(low < high){
            mid = (low + high) / 2;
            if(A[mid] > A[0])
                high = mid - 1;
            else
                low = mid + 1;
        }
        for(j = i - 1; j >= high + 1; --j)
            A[j + 1] = A[j];
        A[high + 1] = A[0];
    }
}
```

```
// ShellSort
/*
    记录前后位置的增量是dk，不是1
    A[0]暂存单元，不是哨兵，当 j<=0 时，插入位置已到
*/
void ShellSort(int A[]){
    int n = A.size();
    for(int dk = n / 2; dk >= 1; dk = dk / 2)
        for(int i = dk + 1; i <= n; ++i)
            if(A[i] < A[i - dk]){
                A[0] = A[i];
                for(int j = i - dk; j > 0 && A[0] < A[j]; j -= dk)
                    A[j + dk] = A[j];
                A[j + dk] = A[0];
            }
}

// 冒泡排序
void BubbleSort(int R[]){
    int n = R.size();
    for(int i = n - 1; i >= 1; --i){
        int flag = 0;
        for(j = 1; j <=i; ++j){
            if(R[j - 1] > R[j]){
                int temp = R[j];
                R[j] = R[j - 1];
                R[j - 1] = temp;
                flag = 1;
            }
        }
        if(flag == 0)
            return;
    }
}
```

```cpp
// 快速排序
#include<iostream>

using namespace std;

const int N = 1000010;
int n;
int arr[N];

void qiuck_sort(int arr[], int l, int r){
    if(l >= r) return;
    int i = l - 1, j = r + 1, x = arr[l + r >> 1];
    while(i < j){
        do i++; while(arr[i] < x);
        do j--; while(arr[j] > x);

        if(i < j) swap(arr[i], arr[j]);
    }
    qiuck_sort(arr, l, j), qiuck_sort(arr, j + 1, r);
}

int main(){
    scanf("%d", &n);

    for(int i = 0; i < n; i++)  scanf("%d", &arr[i]);

    qiuck_sort(arr, 0 , n - 1);

    for(int i = 0; i < n; i++)  printf("%d ", arr[i]);
}
```

```
// 简单选择排序
void SelectSort(int R[], int n){
    int i, j, k;
    int temp;
    for(int = 0; i < n; ++i){
        k = i;
        //从无序列中选择最小的一个关键字
        for(j = i + 1; j < n; ++j){
            if(R[k] > R[j])
                k = j;
        }
        // 最小关键字与无序序列第一个位置交换
        temp = R[i];
        R[i] = R[k];
        R[k] = temp;
    }
}
```

```cpp
// 归并排序
#include<iostream>

using namespace std;

const int N = 1000010;

int n;
int tmp[N];
int q[N];

void merge(int q[], int l, int r){
    if(l >= r) return;

    int mid = l + r >> 1;
    merge(q, l, mid);
    merge(q, mid + 1, r);

    int k = 0, i = l, j = mid + 1;
    while(i <= mid && j <= r){
        if(q[i] < q[j]) tmp[k++] = q[i++];
        else tmp[k++] = q[j++];
    }
    while(i <= mid) tmp[k++] = q[i++];
    while(j <= r) tmp[k++] = q[j++];

    for(i = l, j = 0; i <= r; i++, j++) q[i] = tmp[j];
}

int main(){
    scanf("%d", &n);
    for(int i = 0; i < n; i++){
        scanf("%d", &q[i]);
    }
    merge(q, 0, n - 1);

    for(int i = 0; i < n; i++){
        printf("%d ", q[i]);
    }
}
```

```
// 堆排序
#include<iostream>

using namespace std;

    void swap(int tree[], int a, int b){
        int temp = tree[a];
        tree[a] = tree[b];
        tree[b] = temp;
    }

    void heapify(int tree[], int n, int i){
        int c1 = i * 2 + 1;
        int c2 = i * 2 + 2;
        int max = i;
        if(c1 < n && tree[c1] > tree[max])
            max = c1;
        if(c2 < n && tree[c2] > tree[max])
            max = c2;
        if(max != i){
            swap(tree, max, i);
            heapify(tree, n, max);
        }
    }

    void build_heap(int tree[], int n){
        int last_node = n - 1;
        int parent = (last_node - 1) / 2;
        for(int i = parent; i >= 0; i--){
            heapify(tree, n, i);
        }
    }

    void heapSort(int tree[], int n){
        build_heap(tree, n);
        for(int i = n - 1; i >= 0; i--){
            swap(tree, i, 0);
            heapify(tree, i, 0);
        }
    }

    int main(){
        int n,m;
        scanf("%d%d", &n, &m);
```

```
    int tree[n];
    for(int i = 0; i < n; i++){
        scanf("%d", &tree[i]);
    }
    heapSort(tree, n);
     int i;
    for(i = 0; i < m; i++){
        printf("%d ", tree[i]);
    }
}
```