

二、

- 亂序數組
- 亂序數組排序問題

```
#include<bits/stdc++.h>
using namespace std;
int *getranddata(int n)
{
    int *arr = new int [n];
    for(int i = 0;i < n;i++)
    {
        arr[i] = rand()%1000;
    }
    return arr;
}

bool check(int *arr,int n)
{
    for(int i = 1;i < n;i++)
    {
        if(arr[i] < arr[i - 1])
            return false;
    }
    return true;
}

void print(int *arr ,int n)
{
    for(int i = 0;i < n;i++)
    {
        cout << arr[i] << " ";
    }
    cout << endl;
    return;
}

void selection_sort(int *arr,int n)
{
    for(int i = 0;i < n - 1;i++)
    {
        int ind = i;
        for(int j = i + 1;j < n;j++)
        {
            if(arr[j] < arr[ind]) ind = j;
        }
        swap(arr[i],arr[ind]);
    }
    return;
}
int main()
```

```

{
    int n = 100;
    srand((unsigned)time(NULL));
    int *arr = getranddata(n);
    selection_sort(arr,n);
    print(arr,n);
    cout << check(arr,n) << endl;
    delete arr;
    return 0;
}

```

問題

- 亂数生成
- バブルソート
- セレクションソート

```

#include<bits/stdc++.h>
using namespace std;
int *getranddata(int n)
{
    int *arr = new int [n];
    for(int i = 0;i < n;i++)
    {
        arr[i] = rand()%1000;
    }
    return arr;
}

bool check(int *arr,int n)
{
    for(int i = 1;i < n;i++)
    {
        if(arr[i] < arr[i - 1])
            return false;
    }
    return true;
}

void print(int *arr ,int n)
{
    for(int i = 0;i < n;i++)
    {
        cout << arr[i] << " ";
    }
    cout << endl;
    return;
}

void insert_sort(int *arr,int n)
{

```

```

    for(int i = 1;i < n;i++)
    {
        int j = i;
        while(j > 0 && arr[j - 1] > arr[j])
        {
            swap(arr[j - 1],arr[j]);
            j -= 1;
        }
    }
    return;
}
int main()
{
    int n = 100;
    srand((unsigned)time(NULL));
    int *arr = getranddata(n);
    insert_sort(arr,n);
    print(arr,n);
    cout << check(arr,n) << endl;
    delete arr;
    return 0;
}

```

Algorithm Analysis

- Worst Case Time Complexity (Worst Case Execution Time) $O(n^2)$
- CPU Time Complexity $O(n^2)$
- Space Complexity $O(1)$

Implementation

- Implementation of Insertion Sort using while loop $\text{while}(j > 0)$
- $O(n^2)$
- $O(n)$

```

void unsupervise_insert_sort(int *arr,int n) //Implementation
{
    int ind = 0;
    for(int i = 1;i < n;i++)
    {
        if(arr[ind] > arr[i]) ind = i;
    }
    while(ind > 0)
    {
        swap(arr[ind],arr[ind - 1]);
        ind -= 1;
    }
    for(int i = 1;i < n;i++)
    {
        int j = i;
        while( arr[j - 1] > arr[j])
        {

```

```

        swap(arr[j - 1], arr[j]);
        j -= 1;
    }
}
return;
}

```

Algorithm

- 冒泡排序
- 插入排序
- 归并排序
- 快速排序

$O(n^2)$ (最坏情况) $n/2 \cdot n/4 \cdot n/8 \cdot n/16, \dots O(n^{1.5})$ (最好情况)
 $Hibbard$: 1, 3, 7, ..., 2^{k-1}

Code

```

void shell_sort(int *arr,int n)
{
int k = 2,step;
do
{
    step = n / k == 0 ? 1:n / k;
    for(int i = 0;i < step;i++) //step步长为step的插入排序
        unsupervise_insert_sort(arr,n,step);
    k *= 2;
}while(step != 1);
return;
}

```

Hibrad

```

void shell_sort_hibbard(int *arr,int n)
{
    int step = 1;
    while(step <= n / 2)  step = step * 2 + 1;
    do
    {
        step /= 2;
        for(int i = 0;i < step;i++)
            unsupervise_insert_sort(arr,n,step);
    }while(step > 1);
    return ;
}

```

Complexity

- 最坏情况 cnt 为常数， $break$

```

#include<bits/stdc++.h>
using namespace std;
void bubble_sort(int *arr,int n)
{
    int cnt;
    for(int i = n;i > 0;i--)
    {
        cnt = 0;
        for(int j = 1;j < i;j++)
        {
            if(arr[j] >= arr[j - 1]) continue;
            swap(arr[j],arr[j - 1]);
            cnt += 1;
        }
        if(cnt == 0) break;
    }
    return ;
}

int *getranddata(int n)
{
    int *arr = new int [n];
    for(int i = 0;i < n;i++)
        arr[i] = rand() % 1000;
    return arr;
}

int check(int *arr ,int n)
{
    for(int i = 1;i < n;i++)
        if(arr[i] < arr[i - 1]) return 0;
    return 1;
}

void print(int *arr,int n)
{
    for(int i = 0;i < n;i++)
        cout << arr[i] << " ";
    cout << endl;
    return ;
}
int main()
{
    srand((unsigned)time(NULL));
    int n = 100;
    int *arr = getranddata(n);
    bubble_sort(arr,n);
    print(arr,n);
    cout << check(arr,n) << endl;
    delete arr;
}

```

```
        return 0;
    }
```

問題

- データ構造とアルゴリズム
- データ構造とアルゴリズム
 - データ構造
 - データ構造とアルゴリズム
 - データ構造とアルゴリズムとデータ構造
 - データ構造とアルゴリズムとデータ構造
 - データ構造とアルゴリズムとデータ構造
- データ構造とアルゴリズム(実装)

```
#include<bits/stdc++.h>
using namespace std;

void quick_sort(int *arr ,int l,int r) //問題
{
    if(r - l <= 2) //問題
    {
        if(r - l <= 1) return;
        if(arr[l] > arr[l + 1]) swap(arr[l],arr[l + 1]);
        return ;
    }
    //問題
    int x = l,y = r - 1,z = arr[l];
    while(x < y)
    {
        while(x < y && z <= arr[y]) y--;
        if(x < y) arr[x] = arr[y];
        while(x < y && z >= arr[x]) x++;
        if(x < y) arr[y] = arr[x];
    }
    arr[x] = z;
    quick_sort(arr,l,x);
    quick_sort(arr,x+1,r);
}

int *getranddata(int n)
{
    int *arr = new int [n];
    for(int i = 0;i < n;i++)
        arr[i] = rand() % 1000;
    return arr;
}

int check(int *arr ,int n)
{
    for(int i = 1;i < n;i++)
```

```

        if(arr[i] < arr[i - 1]) return 0;
    return 1;
}

void print(int *arr,int n)
{
    for(int i = 0;i < n;i++)
        cout << arr[i] << " ";
    cout << endl;
    return ;
}

int main()
{
    int n = 100;
    int *arr = getranddata(n);
    quick_sort(arr,0,n);
    print(arr,n);
    cout << check(arr,n) << endl;
    delete arr;
    return 0;
}

```

問題

- v_1
- v_2
- v_3
- v_4

$\text{while}(x \leq y)$ $x = y$ $(l \sim y+1), (x \sim r)$
 $\text{swap}(arr[x++], arr[y--])$ $x++y--$
 $\text{if}(x \leq y)$ $x = y$ $\text{while}(z < arr[y])$ $1\ 1\ 1\ 2\ 2\ 2\ 5\ 4\ 1\ 1\ 1$

```

#include<bits/stdc++.h>
using namespace std;
void unsupervise_insert_sort(int *arr,int l,int r)
{
    int ind = l;
    for(int i = l + 1;i < r;i++)
    {
        if(arr[ind] > arr[i]) ind = i;
    }
    while(ind > l)
    {
        swap(arr[ind],arr[ind - 1]);
        ind -= 1;
    }
}

```

```

    for(int i = l + 1;i < r;i++)
    {
        int j = i;
        while(arr[j - 1] > arr[j])
        {
            swap(arr[j - 1],arr[j]);
            j -= 1;
        }
    }
    return;
}

void print(int *arr,int n)
{
    for(int i = 0;i < n;i++)
        cout << arr[i] << " ";
    cout << endl;
    return ;
}

```

- 亂序

```

void quick_sort(int *arr ,int l,int r) //乱序
{
    if(r - l <= 2) //乱序
    {
        if(r - l <= 1) return;
        if(arr[l] > arr[l + 1]) swap(arr[l],arr[l + 1]);
        return ;
    }
    //乱序
    int x = l,y = r - 1,z = arr[l];
    while(x < y)
    {
        while(x < y && z <= arr[y]) y--;
        if(x < y) arr[x] = arr[y];
        while(x < y && z >= arr[x]) x++;
        if(x < y) arr[y] = arr[x];
    }
    arr[y] = z;
    quick_sort(arr,l,x);
    quick_sort(arr,x+1,r);
}

```

- 乱序乱序乱序乱序

```

void quick_sort_v1(int *arr ,int l,int r) //乱序乱序乱序乱序
{
    if(r - l <= 2) //乱序
    {

```

```

    if(r - l <= 1) return;
    if(arr[l] > arr[l + 1]) swap(arr[l],arr[l + 1]);
    return ;
}
//-----
int x = l,y = r - 1,z = arr[l];
while(x <= y)
{
    while(z < arr[y]) y--;
    while(z > arr[x]) x++;
    if(x <= y)
        swap(arr[x++], arr[y--]);
}
quick_sort_v1(arr,l,x);
quick_sort_v1(arr,x,r);
}

```

- 例題

```

int way(int a,int b,int c)
{
    if(a > b) swap(a,b);
    if(a > c) swap(a,c);
    if(b > c) swap(b,c);
    return b;
}

void quick_sort_v2(int *arr ,int l,int r) //-----
{
    if(r - l <= 2) //-----
    {
        if(r - l <= 1) return;
        if(arr[l] > arr[l + 1]) swap(arr[l],arr[l + 1]);
        return ;
    }
    //-----
    int x = l,y = r - 1;
    int z = way(arr[l],arr[r - 1],arr[(l + r) / 2]);
    while(x < y)
    {
        while( z < arr[y]) y--;
        while( z > arr[x]) x++;
        if(x <= y)
            swap(arr[x++], arr[y--]); //x++ y-- x y -----
        //-----
    }
    quick_sort_v2(arr,l,y+1);
    quick_sort_v2(arr,x,r);
}

```

- 三向切分快速排序

```

void quick_sort_v3(int *arr ,int l,int r) //三向切分快速排序
{
    while(l < r)
    {
        if(r - l <= 2) //小
        {
            if(r - l <= 1) return;
            if(arr[l] > arr[l + 1]) swap(arr[l],arr[l + 1]);
            return ;
        }
        //大
        int x = l,y = r - 1;
        int z = way(arr[l],arr[r - 1],arr[(l + r) / 2]);
        while(x <= y)
        {
            while(z < arr[y]) y--;
            while(z > arr[x]) x++;
            if(x <= y)//三向切分快排
                swap(arr[x++], arr[y--]);
        }
        quick_sort_v3(arr,l,x);
        l = x;
    }
    return ;
}

```

- 四向切分快速排序

```

void __quick_sort_v4(int *arr ,int l,int r) //四向
{
    while(r - l > 16)
    {
        //小
        int x = l,y = r - 1,z = arr[l];
        while(x <= y)
        {
            while(z < arr[y]) y--;
            while(z > arr[x]) x++;
            if(x <= y) //三向快排
                swap(arr[x++], arr[y--]);
        }
        __quick_sort_v4(arr,l,x);
        l = x;
    }
    return ;
}

```

```
void quick_sort_v4(int *arr ,int l,int r) //快速排序
{
    __quick_sort_v4(arr,l,r);
    unsupervise_insert_sort(arr,l,r);
}
```

- x□y□□□□□x□□y□□□□y□□x□□

4


```
void merge_sort(int *arr,int l,int r)
{
    if(r - l <= 1) return ;
    int mid = (l + r) / 2;
    merge_sort(arr,l,mid);
    merge_sort(arr,mid,r);
    int p1 = l , p2 = mid , k = 0;
    while(p1 < mid || p2 < r)
    {
        if(p2 == r || (p1 < mid && arr[p1] <= arr[p2]))
            temp[k++] = arr[p1++];
        else
            temp[k++] = arr[p2++];
    }
    for(int i = l;i < r;i++)
        arr[i] = temp[i - l];
    return ;
}
```

1

10 of 10

- $\Theta(n^2)$ ($\Omega(n^2)$)
 - $\Theta(n^3)$
 - $\Theta(n^4)$
 - $\Theta(n^{\log n})O(n)$

- `temparr`

```

#include<bits/stdc++.h>
using namespace std;

void print(int *arr,int l,int r)
{
    for(int i = l;i < r;i++)
    {
        if(i) cout << " ";
        cout << arr[i];
    }
    cout << endl;
    return;
}

int *getnewdata(int l ,int r)
{
    srand((unsigned)time(NULL));
    int *arr = new int[r - l];
    for(int i = l;i < r;i++)
    {
        arr[i] = rand() % 100;
    }
    return arr;
}
#define K 10
void radix_sort(int *arr,int l,int r)
{
    int *cnt = new int[K];
    int *temp = new int[r - l];
    memset(cnt,0,sizeof(int) * K);
    for(int i = l;i < r;i++) cnt[arr[i] % K] += 1;
    for(int i = 1;i < K;i++) cnt[i] += cnt[i - 1]; //计数
    for(int i = r - 1;i >= l;i--) temp[--cnt[arr[i] % K]] = arr[i];
    memcpy(arr + l,temp,sizeof(int) * (r - l));
    memset(cnt,0,sizeof(int) * K);
    for(int i = l;i < r;i++) cnt[arr[i] / K] += 1;
    for(int i = 1;i < K;i++) cnt[i] += cnt[i - 1]; //计数
    for(int i = r - 1;i >= l;i--) temp[--cnt[arr[i] / K]] = arr[i];
    memcpy(arr + l,temp,sizeof(int) * (r - l));
    delete cnt,temp;
    return ;
}

bool check(int *arr,int l,int r)
{
    for(int i = l + 1;i < r;i++)
    {
        if(arr[i] < arr[i - 1]) return false;
    }
}

```

```

        return true;
    }
#define n 10000
int main()
{
    int *arr = getnewdata(0,n);
    radix_sort(arr,0,n);
    //print(arr,0,n);
    cout << check(arr,0,n);
    delete arr;
    return 0;
}

```

sort()函数

- 第一个参数1表示要排序的数组2表示排序的范围(从到到)
- 第二个
- 第三个参数表示排序的规则3表示重载greater<>()
- 第四个参数vector的end()表示排序结束的标志

示例代码

1. 整数排序

```

bool cmp(int a,int b)
{
    return a > b; //如果a>b返回true否则false
}
sort(arr,arr + 10;cmp);

```

2. 结构体排序

```

bool cmp(struct a,struct b)
{
    if(a.x != b.x) return a.x > b.x;
    return a.y < b.y;
}

```

- 结构体x表示y表示

注意



- 第一个参数表示要排序的数组
- 第二个参数表示排序的范围(从到到)
- 第三个参数表示排序的规则(重载greater<>())
 - target表示p1和p2
 - arr表示

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        vector<int> arr,ans;
        for(int i = 0;i < nums.size();i++)
            ans.push_back(i);
        sort(ans.begin(),ans.end(),[&](int i,int j) ->bool
        {
            return nums[i] < nums[j];
        });
        int p1 = 0,p2 = nums.size() - 1;
        while(nums[ans[p1]] + nums[ans[p2]] != target)
        {
            if(nums[ans[p1]] + nums[ans[p2]] > target)
                p2--;
            if(nums[ans[p1]] + nums[ans[p2]] < target)
                p1++;
        }
        arr.push_back(ans[p1]);
        arr.push_back(ans[p2]);
        return arr;
    }
};
```

1



- $\text{z}^{\text{max}}(\text{z}^{\text{max}}\text{z}^{\text{max}}\text{z}^{\text{max}}\text{z}^{\text{max}}\text{z}^{\text{max}}\text{z}^{\text{max}})$
 - $z^{\text{max}}(z^{\text{max}})(z^{\text{max}})(z^{\text{max}})(z^{\text{max}})$

```
class Solution {
public:
    ListNode* sortList(ListNode* head) {
        if(head == NULL || head->next == NULL) return head;
        int l = head->val, r = head->val, z;
        ListNode *p = head,*q,*h1 = NULL,*h2 = NULL;
        while(p) l = min(p->val,l), r = max(p->val,r),p = p->next;
        z = (l + r) >> 1;
        if(l == r) return head;
        p = head;
        while(p)
        {
            q = p->next;
            if(p->val <= z)
            {
                p->next = h1;
                h1 = p;
            }
        }
    }
};
```

```

    else
    {
        p->next = h2;
        h2 = p;
    }
    p = q;
}
h1 = sortList(h1);
h2 = sortList(h2);
p = h1;
while(p->next) p = p->next;
p->next = h2;
return h1;
}
};

```

- 二分法

```

class Solution {
public:
    int getlength(ListNode *head)
    {
        int n = 0;
        while(head) n+=1 , head = head->next;
        return n;
    }
    ListNode *merge_sort(ListNode *head , int n)
    {
        if(n <= 1) return head;
        int l = n / 2 , r = n - l;
        ListNode *p = head,*p1 = head,*p2,new_head;
        for(int i = 1;i < l;i++) p = p->next;
        p2 = p->next;
        p->next = NULL;
        p1 = merge_sort(p1,l);
        p2 = merge_sort(p2,r);
        p = &new_head,new_head.next = NULL;
        while(p1 || p2)
        {
            if(p2 == NULL || (p1 && p1->val < p2->val))
            {
                p->next = p1;
                p = p1;
                p1 = p1->next;
            }
            else
            {
                p->next = p2;
                p = p2;
                p2 = p->next;
            }
        }
    }
};

```

```

    }
    return new_head.next;
}
ListNode* sortList(ListNode* head) {
    int n = getlength(head);
    return merge_sort(head, n);
}
};

```

二分法

- 二分法

```

class Solution {
public:
    int t = 65536;
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        if(n == 0) return;
        for(int i = 0;i < n;i++) nums1[m + i] = nums2[i];
        sort(nums1.begin(),nums1.end());
        return;
    }
};

```

- 二分法

```

class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
vector<int> arr;
arr.resize(nums1.size());
int p1 = 0 , p2 = 0,k = 0;
while(p1 != m || p2 != n )
{
    if(p2 == n || (p1 != m && nums1[p1] < nums2[p2]))
    {
        arr[k] = nums1[p1];
        p1++,k++;
    }
    else
    {
        arr[k] = nums2[p2];
        p2++,k++;
    }
}
for(int i = 0;i <(m + n);i++)
{
    nums1[i] = arr[i];
}
return ;

```

};

- `int[]`
 - `int[] nums1` `int[] nums2`

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        int p1 = m - 1, p2 = n - 1, k = m + n - 1;
        while(p1 >= 0 || p2 >= 0)
        {
            if(p2 == -1 || (p1 != -1 && nums1[p1] > nums2[p2]))
            {
                nums1[k] = nums1[p1];
                p1--, k--;
            }
            else
            {
                nums1[k] = nums2[p2];
                p2--, k--;
            }
        }
        return ;
    }
};
```

1

- 

```
class Solution { public: ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) { ListNode *p, *new_head; p = &new_head; while(list1 || list2) { if(list2 == NULL || (list1 != NULL && list1->val < list2->val)) { p->next = list1; list1 = list1->next; p = p->next; } else { p->next = list2; list2 = list2->next; p = p->next; } } return new_head.next; } };
```

□□□□□□□□□□



```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        int n = nums1.size(), m = nums2.size();
        vector<int> temp(n + m);
        int p1 = 0, p2 = 0, k = 0;
        while(p1 != n || p2 != m)
        {
            if(p2 == m || (p1 != n && nums1[p1] < nums2[p2]))

```

```

    {
        temp[k] = nums1[p1];
        p1++, k++;
    }
    else
    {
        temp[k] = nums2[p2];
        p2++, k++;
    }
}
int t = temp.size() / 2;
if(temp.size() % 2)
    return temp[t];
return (temp[t] + temp[t - 1]) / 2.0;
}
};


```

□□□□□

- □□

```

class Solution {
public:
    bool containsNearbyDuplicate(vector<int>& nums, int k) {
        int p1 = 0, p2 = 1, n = nums.size();
        while (p1 < n) {
            while (p2 < n && (p2 - p1) <= k) {
                if (nums[p1] == nums[p2])
                    return true;
                p2++;
            }
            p1++;
            p2 = p1 + 1;
        }
        return false;
    }
};


```

* □□

```

class Solution {
public:
    bool containsNearbyDuplicate(vector<int>& nums, int k) {
        int n = nums.size();
        vector<int> ind(n);
        for(int i = 0; i < n; i++) ind[i] = i;
        sort(ind.begin(),ind.end(),[&](int i ,int j)->bool
        {
            return nums[i] < nums[j];
        });
        for(int i = 0;i < n - 1;i++)

```

```

    {
        if(nums[ind[i]] != nums[ind[i + 1]]) continue;
        if(abs(ind[i] - ind[i + 1]) <= k) return true;
    }
    return false;
}
};

```

問題文



- a(0,1,2,3,4),b(5,6,7,8,9),c(10,11,12,13,14)
- a,b,c
- a,b,c
 a,b,c
- p1,p2,temp,p1,a,p2
- temp, arr

```

#include<bits/stdc++.h>
using namespace std;
int arr[500005],temp[500005];

long long merge_sort(int *arr,int l,int r)
{
    if(r - l <= 1) return 0;
    int mid = (r + l) / 2;
    long long a = merge_sort(arr, l, mid);
    long long b = merge_sort(arr, mid, r);
    long long c = 0;
    int p1 = l, p2 = mid, k = 0;
    while(p1 != mid || p2 != r)
    {
        if(p2 == r || (p1 != mid && arr[p1] <= arr[p2]))
            temp[k++] = arr[p1++];
        else
        {
            temp[k++] = arr[p2++];
            c += (mid - p1);
        }
    }
    for(int i = l;i < r;i++) arr[i] = temp[i - l]; //合併
    return a + b + c;
}
void solve(int n)
{
    for(int i = 0;i < n;i++) cin >> arr[i];
    cout << merge_sort(arr,0,n) << endl;
    return;
}

```

```
}

int main()
{
    int n;
    while(1)
    {
        cin >> n;
        if(n == 0) break;
        solve(n);
    }
    return 0;
}
```

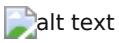
2



- $\prod_{i=1}^n y_i$
 - $y \sum_{i=1}^n \$\$$
 - 
 - $x \sum_{i=1}^n \$\$$
 - $\|x - x_i\|$
 - $\prod_{i=1}^n$

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin >> n;
    vector<int> x(n),y(n);
    for(int i = 0;i < n; i++) cin >> x[i] >> y[i];
    int X,Y,costx = 0,costy = 0;
    sort(x.begin(),x.end());
    for(int i = 0; i < n; i++) x[i] = x[i] - i;
    sort(x.begin(),x.end());
    sort(y.begin(),y.end());
    X = x[n / 2];
    Y = y[n / 2];
    for(int i = 0;i < n; i++) costy += abs(y[i] - Y);
    for(int i = 0;i < n; i++) costx += abs(x[i] - X);
    cout << costx + costly << endl;
    return 0;
}
```

1



- 1
 - \$C_{i+1}\$
 - 21

```


$$\dots C_{\{i+1\}} \dots 1 \dots C_{\{i+1\}}^{\wedge \{1\}} \dots C_{\{i+1\}} \dots 2 \dots 1 \dots$$


$$\dots C_{\{i+1\}} \dots C_{\{i\}}^{\wedge \{1\}} \dots A_{\{i\}} * B_i \backslash \text{eq } A_{\{i+1\}} * V_{\{i+1\}} \dots$$


$$\dots A_{\{i\}} * B_i \backslash \text{eq } A_{\{i+1\}} * V_{\{i+1\}} \dots \$C_i \dots 1 \dots$$


```

2020-07-01