20220929 数据结构与算法 解题报告

明明的随机数

Solution #1

题目已给出条件：所有出现的数字在(1,1000)之间。可利用该特性使用桶排序。

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

int main() {
    for (int n; cin >> n; ) {
        int m = n;
        vector<int> buc(1001, 0);
        for (int i = 0; i < n; i++) {
            int tmp; cin >> tmp;
            if (buc[tmp]) m--;
            buc[tmp] = 1;
        }
        cout << m << endl;
        for (int i = 1; i <= 1000; i++)
            if (buc[i])
                cout << i << " ";
        cout << endl;
    }
    return 0;
}
```

Solution #2

利用C++的sort，unique函数完成排序去重。

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

int main(int argc, char const *argv[]) {
  // freopen("init.in", "r", stdin);
  for (int n; cin >> n;) {
    vector<int> val(n);
    for (auto &i : val) cin >> i;
    sort(val.begin(), val.end());
    auto it = unique(val.begin(), val.end());
    cout << it - val.begin() << endl;
    for (auto cur = val.begin(); cur != it; cur++) cout << *cur
<< " ";
    cout << endl;
  }
  return 0;
}
```

INVERSION NUMBER

利用归并排序，在每次归并操作时统计答案。

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

#define LL long long

LL mergeSort(vector<int> &val, int l, int r) {
  if (l + 1 == r) return 0;
  int mid = l + r >> 1;
  LL ans = mergeSort(val, l, mid) + mergeSort(val, mid, r);
  vector<int> tmp(r - l);
  int p = l, q = mid;
  for (int i = 0; i < r - l; i++) {
```

```
    if (p != mid && (q == r || val[p] <= val[q])) {tmp[i] =
val[p]; ans += r - q; p++;}
    else {tmp[i] = val[q]; q++;}
  }
  for (int i = l; i < r; i++)
    val[i] = tmp[i - l];
  return ans;
}

int main() {
  // freopen("init.in", "r", stdin);
  LL n;
    for (; cin >> n;) {
        vector<int> val(n); for (int i = 0; i < n; i++) cin >>
val[i];
        cout << n * (n - 1LL) / 2LL - mergeSort(val, 0, n) <<
endl;
    }
  return 0;
}
```

MERGESORT OF LIST

链表上的归并排序，需要注意的是指针的处理问题，可通过每次归并操作结束后子链尾指针赋值为空很好的解决。

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

void mergesort(linkedlist *&head, int len) {
    if (len == 1) {head->next = NULL; return;}
    linkedlist *p = head, *q = head;
    for (int i = 0; i < len / 2; i++)
        q = q->next;
    mergesort(p, len / 2), mergesort(q, len - len / 2);
    if (p->data < q->data) {head = p; p = p->next;}
    else {head = q; q = q->next;}
```

```
    for (linkedlist *cur = head; p != NULL || q != NULL; cur =
cur->next) {
        if (p == NULL) {cur->next = q; q = q->next;}
        else if (q == NULL) {cur->next = p; p = p->next;}
        else if (p->data < q->data) {cur->next = p; p = p-
>next;}
        else {cur->next = q; q = q->next;}
    }
}
```