Busca e Ordenação

Ordenação em C/C++: problemas resolvidos

Prof. Edson Alves

2019

Faculdade UnB Gama

Sumário

URI 1259 – Pares e Ímpares

Problema

Considerando a entrada de valores inteiros não negativos, ordene estes valores segundo o seguinte critério:

- Primeiro os Pares
- Depois os Ímpares

Sendo que deverão ser apresentados os pares em ordem crescente e depois os ímpares em ordem decrescente.

Entrada e saída

Entrada

A primeira linha de entrada contém um único inteiro positivo N $(1 < N < 10^5)$. Este é o número de linhas de entrada que vem logo a seguir. As próximas N linhas conterão, cada uma delas, um valor inteiro não negativo.

Saída

Apresente todos os valores lidos na entrada segundo a ordem apresentada acima. Cada número deve ser impresso em uma linha, conforme exemplo abaixo.

Exemplo de entradas e saídas

Exemplo de Entrada

```
10
4
32
34
543
3456
654
567
87
6789
98
```

Exemplo de Saída

```
4
32
34
98
654
3456
6789
567
543
87
```

Solução com complexidade $O(N \log N)$

- O problema pode ser resolvido armazenando-se os pares e ímpares em vetores distintos, e aplicando a ordenação correspondente em cada um destes vetores
- Contudo, é possível ordenar todo o vetor de uma só vez, através da escrita de um comparador customizado
- A paridade de um número n pode ser obtida através do resto da divisão por 2
- Se a e b tem paridades distintas, a será menor do que b se for par: logo basta comparar as paridades (pois zero significa par, um significa ímpar)
- Se a e b tem mesma paridade, a comparação no caso par é $a < b$; no caso ímpar, $a > b$
- Por conta da chamada da função `sort()`, o algoritmo tem complexidade $O(N \log N)$

Solução $O(N \log N)$

```cpp
#include <bits/stdc++.h>

using namespace std;

void solve(vector<int>& ns)
{
    sort(ns.begin(), ns.end(), [](int a, int b)
        {
            int pa = a % 2, pb = b % 2;

            return pa == pb ? (pa ? a > b : a < b) : pa < pb;
        }
    );
}

```

Solução $O(N \log N)$

```
16 int main()
17 {
18     int N;
19
20     while (cin >> N)
21     {
22         vector<int> ns(N);
23
24         for (int i = 0; i < N; i++)
25             cin >> ns[i];
26
27         solve(ns);
28
29         for (const auto& n : ns)
30             cout << n << '\n';
31     }
32
33     return 0;
34 }
```

UVA 11100 – The Trip, 2007

Problema

A number of students are members of a club that travels annually to exotic locations. Their destinations in the past have included Indianapolis, Phoenix, Nashville, Philadelphia, San Jose, Atlanta, Eindhoven, Orlando, Vancouver, Honolulu, Beverly Hills, Prague, Shanghai, and San Antonio. This spring they are hoping to make a similar trip but aren't quite sure where or when.

An issue with the trip is that their very generous sponsors always give them various knapsacks and other carrying bags that they must pack for their trip home. As the airline allows only so many pieces of luggage, they decide to pool their gifts and to pack one bag within another so as to minimize the total number of pieces they must carry.

Problema

The bags are all exactly the same shape and differ only in their linear dimension which is a positive integer not exceeding 1000000. A bag with smaller dimension will fit in one with larger dimension. You are to compute which bags to pack within which others so as to minimize the overall number of pieces of luggage (i.e. the number of outermost bags). While maintaining the minimal number of pieces you are also to minimize the total number of bags in any one piece that must be carried.

Entrada e saída

Input

Standard input contains several test cases. Each test case consists of an integer $1 \leq n \leq 10000$ giving the number of bags followed by n integers on one or more lines, each giving the dimension of a piece. A line containing 0 follows the last test case.

Output

For each test case your output should consist of k, the minimum number of pieces, followed by k lines, each giving the dimensions of the bags comprising one piece, separated by spaces . Each dimension in the input should appear exactly once in the output, and the bags in each piece must fit nested one within another. If there is more than one solution, any will do. Output an empty line between cases.

Exemplo de entradas e saídas

Exemplo de Entrada

```
6
1 1 2 2 2 3
0
```

Exemplo de Saída

```
3
1 2
1 2
3 2
```

Solução $O(N \log N)$

- O principal ponto a ser observado é que objetos com a mesma dimensão devem ir em bolsas distintas
- Logo, o número mínimo de bolsas necessárias é igual ao número de ocorrências n da dimensão x mais frequente
- Alocado o espaço para as bolsas, basta distribuir uniformemente os demais objetos entre as bolsas, do maior para o menor
- Esta distribuição garante que há, no máximo, um elemento com dimensão x em cada bolsa, e que as bolsas estarão ordenadas, do maior para o menor
- Por conta da ordenação utilizada na construção do histograma, esta solução tem complexidade $O(N \log N)$

Solução $O(N \log N)$

```cpp
#include <bits/stdc++.h>

using namespace std;

vector<vector<int>> solve(vector<int>& ds)
{
    map<int, int> hist;
    int n = 0;

    for (const auto& d : ds)
    {
        hist[d]++;
        n = max(n, hist[d]);
    }

    sort(ds.begin(), ds.end(), greater<int>());

    vector<vector<int>> bags(n);

    int pos = 0;

```

Solução $O(N \log N)$

```
22      for (const auto& d : ds)
23          bags[pos++ % n].push_back(d);
24
25      return bags;
26  }
27
28  int main()
29  {
30      ios::sync_with_stdio(false);
31
32      int N, test = 0;
33
34      while (cin >> N, N)
35      {
36          vector<int> ds(N);
37
38          for (int i = 0; i < N; ++i)
39              cin >> ds[i];
40
41          auto ans = solve(ds);
42
```

Solução $O(N \log N)$

```
43          if (test++)
44              cout << endl;
45
46          cout << ans.size() << endl;
47
48          for (size_t i = 0; i < ans.size(); ++i)
49              for (size_t j = 0; j < ans[i].size(); ++j)
50                  cout << ans[i][j] << (j + 1 == ans[i].size() ? "\n": " ");
51      }
52
53      return 0;
54  }
```

Referências

1. URI 1259 – Pares e Ímpares
2. UVA 11100 – The Trip, 2007