Codeforces Round #150 (Div. 2)

Problema A: *Dividing Orange*

Prof. Edson Alves – UnB/FCTE

Problema

One day Ms Swan bought an orange in a shop. The orange consisted of $n \cdot k$ segments, numbered with integers from 1 to $n \cdot k$.

There were k children waiting for Ms Swan at home. The children have recently learned about the orange and they decided to divide it between them. For that each child took a piece of paper and wrote the number of the segment that he would like to get: the i-th ($1 \leq i \leq k$) child wrote the number a_i ($1 \leq a_i \leq n \cdot k$). All numbers a_i accidentally turned out to be different.

Problema

Now the children wonder, how to divide the orange so as to meet these conditions:

- each child gets exactly n orange segments;
- the i-th child gets the segment with number a_i for sure;
- no segment goes to two children simultaneously.

Help the children, divide the orange and fulfill the requirements, described above.

Entrada e saída

Input

The first line contains two integers n, k $(1 \leq n, k \leq 30)$. The second line contains k space-separated integers $a_1, a_2, \ldots, a_k$ $(1 \leq a_i \leq n \cdot k)$, where a_i is the number of the orange segment that the i-th child would like to get.

It is guaranteed that all numbers a_i are distinct.

Output

Print exactly $n \cdot k$ distinct integers. The first n integers represent the indexes of the segments the first child will get, the second n integers represent the indexes of the segments the second child will get, and so on. Separate the printed numbers with whitespaces.

You can print a child's segment indexes in any order. It is guaranteed that the answer always exists. If there are multiple correct answers, print any of them.

Exemplo de entradas e saídas

Sample Input

```
2 2
4 1

3 1
2
```

Sample Output

```
2 4
1 3

3 2 1
```

Solução com complexidade $O(N)$

- Este problema pode ser resolvido com complexidade linear por meio do uso de um `unordered_set` S
- Este conjunto representa os segmentos da laranja que já foram atribuídos a uma das crianças
- Inicialmente deve ser inseridos em S todos os valores a_i
- Como o `unordered_set` utiliza *hashes* em sua implementação, cada consulta ou inserção em S tem complexidade média $O(1)$
- Em seguida, inicializa-se uma variável i apontando para o próximo segmento a ser considerado
- Assim, para cada uma das crianças, avalia-se se i está ou não disponível
- Se estiver disponível, i é atribuído à criança e acrescido em S
- Quando uma criança tiver N segmentos o processamento para a criança seguinte
- Como cada segmento será processado uma única vez, a complexidade da solução é $O(N)$

Solução AC com complexidade $O(N)$

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

using namespace std;

vector<vector<int>> solve(size_t N, int K, const vector<int>& as)
{
    unordered_set<int> used(as.begin(), as.end());
    vector<vector<int>> ans(K);
    int nxt = 1;

    for (int i = 0; i < K; ++i)
    {
        ans[i].push_back(as[i]);

        while (ans[i].size() < N) {
            if (used.count(nxt) == 0)
            {
                ans[i].push_back(nxt);
                used.insert(nxt);
            }
```

Solução AC com complexidade $O(N)$

```
22              ++nxt;
23          }
24      }
25
26      return ans;
27  }
28
29  int main()
30  {
31      ios::sync_with_stdio(false);
32
33      int N, K;
34      cin >> N >> K;
35
36      vector<int> as(K);
37
38      for (int i = 0; i < K; ++i)
39          cin >> as[i];
40
41      auto ans = solve(N, K, as);
```

Solução AC com complexidade $O(N)$

```
43      for (const auto& xs : ans)
44          for (int i = 0; i < N; ++i)
45              cout << xs[i] << (i + 1 == N ? '\n' : ' ');
46
47      return 0;
48  }
```