OBI 2024 - Nível 2: Fase 1

Concurso

Prof. Edson Alves

Faculdade UnB Gama

Cláudia trabalha na OBI (Organização dos Bons Informáticos), que recentemente realizou um concurso para contratar novos funcionários. Agora, Cláudia tem a tarefa de determinar a *nota de corte* para o concurso. Chamamos de nota de corte a nota mínima necessária para ser aprovado no concurso. Ou seja, se a nota de corte do concurso for C, então todos os participantes com uma nota maior ou igual a C serão aprovados no concurso e todos com nota menor que C serão reprovados.

Seu chefe pediu para que Cláudia aprove no mínimo K candidatos do concurso para a próxima fase, mas ela também não quer que a nota de corte seja muito baixa. Por isso, Cláudia decidiu que a nota de corte deverá ser a maior nota C que faz com que no mínimo K candidatos sejam aprovados.

Sua tarefa é: dados o número N de candidatos, as notas $A_1, A_2, \ldots, A_N$ dos candidatos e a quantidade mínima de aprovados K, diga qual deve ser a maior nota de corte C para que pelo menos K candidatos sejam aprovados.

Entrada

A primeira linha da entrada contém dois inteiros, N e K, representando, respectivamente, o número de participantes e o número mínimo de candidatos que devem ser aprovados.

A segunda linha da entrada contém N inteiros A_i, representando as notas dos participantes.

Saída

Seu programa deve imprimir uma linha contendo um único inteiro C, a nota de corte que deve ser escolhida por Cláudia.

Restrições

- $1 \leq K \leq N \leq 500$
- $1 \leq A_i \leq 100$ para todo $1 \leq i \leq N$

Informações sobre a pontuação

A tarefa vale 100 pontos. Estes pontos estão distribuídos em subtarefas, cada uma com suas **restrições adicionais** às definidas acima.

- **Subtarefa 1 (0 pontos):** Esta subtarefa é composta apenas pelos exemplos mostrados abaixo. Ela não vale pontos, serve apenas para que você verifique se o seu programa imprime o resultado correto para os exemplos.
- **Subtarefa 2 (20 pontos):** $K = 1$.
- **Subtarefa 3 (20 pontos):** $K = 3$.
- **Subtarefa 4 (20 pontos):** $A_i \leq 2$.
- **Subtarefa 5 (40 pontos):** Sem restrições adicionais.

Exemplo de entrada e saída

Exemplo de entrada e saída

```
8 9
```

Solução

Solução

- Sandro será capaz de cumprir suas tarefas apenas se elas puderem ser ordenadas de tal forma que as prioridades sejam respeitadas

Solução

- Sandro será capaz de cumprir suas tarefas apenas se elas puderem ser ordenadas de tal forma que as prioridades sejam respeitadas

- Em outras palavras, há solução apenas se existe uma ordenação topológica

Solução

- Sandro será capaz de cumprir suas tarefas apenas se elas puderem ser ordenadas de tal forma que as prioridades sejam respeitadas

- Em outras palavras, há solução apenas se existe uma ordenação topológica

- Se o grafo tem um ou mais ciclos, a resposta é `Sandro fails.`

Solução

- Sandro será capaz de cumprir suas tarefas apenas se elas puderem ser ordenadas de tal forma que as prioridades sejam respeitadas
- Em outras palavras, há solução apenas se existe uma ordenação topológica
- Se o grafo tem um ou mais ciclos, a resposta é `Sandro fails.`
- O problema pede, na saída, uma ordenação específica

Solução

⋆ Sandro será capaz de cumprir suas tarefas apenas se elas puderem ser ordenadas de tal forma que as prioridades sejam respeitadas

⋆ Em outras palavras, há solução apenas se existe uma ordenação topológica

⋆ Se o grafo tem um ou mais ciclos, a resposta é `Sandro fails.`

⋆ O problema pede, na saída, uma ordenação específica

⋆ Esta ordenação pode ser obtida se a fila do algoritmo de Kahn for substituída por uma *min heap*

```cpp
vector<int> topological_sort(int N)
{
    vector<int> o;
    priority_queue<int, vector<int>, greater<int>> q;

    for (int u = 1; u <= N; ++u)
        if (in[u].empty())
            q.push(u);

    while (not q.empty())
    {
        auto u = q.top();
        q.pop();

        o.emplace_back(u);

        for (auto v : out[u])
        {
            in[v].erase(u);
```

```
            if (in[v].empty())
                q.push(v);
        }
    }

    return (int) o.size() == N ? o : vector<int> { };
}

vector<int> solve(int N)
{
    return topological_sort(N);
}
```