Cubo Interdimensional

Kaio é um viajante espacial com um dispositivo especial que o permite transicionar entre dimensões, mas mesmo com essa habilidade ele está atrasado para a escola! Pior ainda, ele está na dimensão mais distante possível da escola e precisa encontrar uma rota para a aula.

O multiverso de Kaio funciona da seguinte forma: cada dimensão é identificada por uma string binária de tamanho **n**, existindo $\mathbf{2^n}$ dimensões distintas. Além disso, duas dimensões são **vizinhas se suas strings binárias diferem em exatamente uma posição**. Seu dispositivo de viagem interdimensional te permite viajar somente entre dimensões vizinhas.

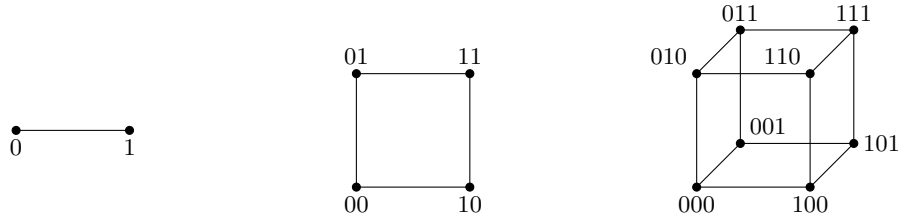

Figure 1: Exemplo de multiverso para $n = 1, 2, 3$. Aqui os rótulos não foram embaralhados.

Kaio está na dimensão $(00\ldots0)$, a escola está na dimensão $(11\ldots1)$ e deseja encontrar um caminho que conecta as duas dimensões. Porém, o dispositivo de locomoção interdimensional está danificado e **os identificadores das dimensões estão todos embaralhados**. Kaio já havia salvo a dimensão T da escola e sabe a dimensão S em que se encontra.

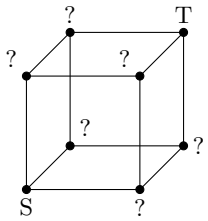

Figure 2: Situação inicial de Kaio.

Para encontrar o caminho que conecta os dois, Kaio pode acessar os dados do dispositivo e descobrir **quais são as dimensões vizinhas** de qualquer outra. Porém, por um limite de bateria ele só pode descobrir os vizinhos de no máximo **2500** dimensões.

Sendo mais preciso, a operação que ele pode realizar com seu dispositivo é a seguinte:

- `vizinhanca(string v)`: descobre a vizinhança da dimensão `v`, recebendo uma lista das n dimensões vizinhas. É necessário que v seja uma string binária de tamanho n.

Ajude Kaio a chegar até a escola contando para ele quais dimensões ele deve perguntar a vizinhança e encontrando um caminho que o liga até a escola!

Detalhes da interação com o corretor

Você pode usar o arquivo **cubo.cpp** como base para começar sua implementação.

Você deve implementar a função:

- `vector<string> encontra_caminho(int n, string S, string T)`
  - **Argumentos:** Um inteiro n representando o tamanho das strings das dimensões, uma string S correspondendo à dimensão inicial e uma string T correspondendo à dimensão final;
  - **Valor de retorno:** Um vetor de strings $[v_1, v_2, \ldots, v_m]$ correspondendo ao caminho de S para T. O caminho deve começar em S, terminar em T e duas dimensões consecutivos precisam ser vizinhas.

Vale ressaltar que o **juiz não é adaptativo**! Ao inicio do programa os rótulos das dimensões são embaralhados e eles não se alteram.

Testes Locais

Para testar seu programa localmente, você deve usar o arquivo `corretor_local.cpp` fornecido. Nesse caso, recomendamos que, ao invés de copiar e colar o conteúdo desse arquivo, você compile os dois arquivos conjuntamente

Exemplo: `g++ -o cubo -O2 cubo.cpp corretor_local.cpp` para compilar e `./cubo` para rodar o testador. Para Windows, adicionamos um arquivo `compila-1.bat` que executa exatamente a linha acima, sem a necessidade do uso do terminal.

O corretor local possui uma interface guiada, a seguir está como ele deveria se comportar para o exemplo abaixo. Você deve executar o corretor local da seguinte forma:

Tamanho das strings (n): **3**
Vizinhanca da dimensao **010**: 011 100 101
Vizinhanca da dimensao **101**: 111 001 010
Caminho encontrado em 2 perguntas
010 100 001 101 111 110

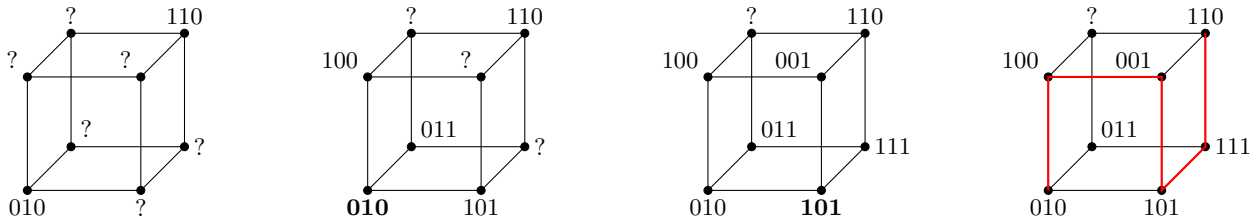

Figure 3: Exemplo de interação.

Perceba que este é apenas um exemplo de como utilizar o corretor local, e não tem nenhuma obrigação de representar uma estratégia que de fato resolva o problema. O corretor irá mostrar todos os valores retorno de `vizinhanca()`.

Restrições

- Seja n como descrito no enunciado, $1 \leq n \leq 50$, o multiverso possui 2^n dimensões e cada dimensão corresponde a uma string binária de tamanho n.
- O número de chamadas de `vizinhanca()` deve ser no máximo **2500**.

Parciais

Seja `v` a string original que representa uma dimensão e `p(v)` o novo rótulo dessa dimensão:

- **Subtask 1** (9 pontos): O cubo não foi embaralhado, isto é, $\texttt{v} = \texttt{p(v)}$.
- **Subtask 2** (13 pontos): O cubo foi embaralhado, mas as strings diferem em no máximo uma posição, isto é, $|\texttt{v} - \texttt{p(v)}| \leq 1$.
- **Subtask 3** (13 pontos): $n \leq 11$.
- **Subtask 4** (27 pontos): $n \leq 20$.
- **Subtask 5** (38 pontos): $n \leq 50$.