- Problema A - Tri-du

  A probabilidade de ganhar a partida é maximizada quando você pega outra cópia da carta de maior valor.

- Problema B - Marcus

  Implemente um *backtracking* que usa como estado a posição atual na matriz e o próximo caractere a ser encontrado.

- Problema C - Encolhendo Polígonos

  Para cada divisor d do perímetro p do círculo, verifique se existem d vértices separados por distâncias de p/d unidades de comprimento de arco. Se existem, essa é uma forma de "encolher" o polígono, agora basta escolher a forma que passa por mais vértices.

- Problema D - Primo

  Os testes do juiz para esse problema são muito fracos!!! Se você quer saber se merecia ter acertado, rode seu programa tendo $N = 2147483647$ como entrada, verifique se a resposta dada é "sim" e se seu programa termina em menos de um segundo. Muitos teriam tomado o veredito de Tempo Limite Excedido e, alguns, de Resposta Errada se esse teste fosse usado.

  O enunciado do problema diz que N pode ser tão grande quanto $2^{31} - 1$ e, então, algoritmos lineares não deveriam ser rápidos o suficiente. Para tornar a solução mais eficiente, basta notar que, se N não tem nenhum divisor positivo d, com $1 < d \leq \sqrt{N}$, ele só poderá ter um divisor maior que $\sqrt{N}$, que é exatamente $N/1 = N$. Então, você precisaria verificar a divisibilidade de N por apenas $\mathcal{O}(\sqrt{N})$ inteiros para saber se ele é primo ou não.

- Problema E - Grid

  Note que, se Bob joga de forma ótima, as escolhas de Alice não interferem no resultado do jogo, porque ela vai acabar tendo riscado todas as linhas da matriz, de qualquer forma. Então, sua única tarefa é avaliar todas as possibilidades de jogadas que Bob pode fazer, que são todas as formas de escolher as células de forma que exista exatamente uma escolhida em cada linha e em cada coluna. A função `next_permutation()` do C++ torna a implementação trivial.

- Problema F - Dama

  Se a dama estiver na casa de destino, a resposta é 0. Se estiver na mesma linha ou coluna ou diagonal da casa de destino, a resposta é 1. Caso

contrário, ela sempre pode usar um movimento para chegar na coluna desejada e mais um para alcançar a casa de destino, ou seja, a resposta é 2.

- Problema G - Produto do Intervalo

  Use duas árvores de segmentos, uma para contar a quantidade de números negativos no intervalo dado e a outra para contar a quantidade de zeros nesse intervalo. Se a quantidade de zeros for positiva em um intervalo, o produto dos números é zero. Caso contrário, o produto é positivo se a quantidade de números negativos é par e é negativo se essa quantidade é ímpar.

- Problema H - Homework

  Se, para um dado N, soubermos contar quantas soluções inteiras e não negativas tem a equação $a+b+c = N$, o problema se torna trivial. Vamos, então, tentar responder uma pergunta um pouco mais geral, usando dois métodos diferentes: quantas soluções tem a equação

  $$x_1 + x_2 + \cdots + x_m = n,$$

  em que as variáveis x_i, $1 \leq i \leq m$, assumem valores inteiros não negativos?

  Método 1: combinatória. Imagine que tenhamos uma fila de n bolinhas e elas devem ser separadas em m compartimentos usando $m-1$ barras verticais. Podemos fazer isso usando qualquer uma das $\frac{(n+m-1)!}{n!(m-1)!} = \binom{n+m-1}{n}$ permutações dessa configuração, porque temos $n+m-1$ objetos no total, sendo n bolinhas e $m-1$ barras. Esse problema é exatamente equivalente ao de somar n unidades usando m parcelas de inteiros não negativos e, portanto, concluímos que a resposta para nossa pergunta inicial também é $\binom{n+m-1}{n}$. Para ficar mais claro, considere o exemplo em que $n = 6$ e $m = 4$. Uma das configurações possíveis para que 4 variáveis somem 6 é $2 + 3 + 0 + 1 = 6$, o que seria representado pelo seguinte esquema:

  $$\circ\circ \,\big|\, \circ\circ\circ \,\big|\big|\, \circ$$

  Todas as outras soluções de $x_1 + x_2 + x_3 + x_4 = 6$ podem ser obtidas permutando esses $6 + 3 = 9$ objetos.

  Método 2: programação dinâmica. Se fixarmos uma das m variáveis em um valor k, caímos no subproblema de somar $n-k$ usando $m-1$ variáveis. Então, definindo $f(n, m)$ como sendo o número de soluções inteiras e não negativas de uma equação de m variáveis que somam n, temos a seguinte relação de recorrência:

  $$f(n, m) = \sum_{k=0}^{n} f(n-k, m-1),$$

o que, combinado a uma matriz de memoização, fornece uma solução usando memória $\mathcal{O}(n)$ e tempo $\mathcal{O}(n^2 m)$.

- Problema I - Grid de Largada

  Como a entrada é muito pequena, não precisamos usar nenhuma técnica sofisticada, basta simular todas as ultrapassagens.

- Problema J - Máquina Dobradora

  Use *backtracking* para fazer todas as dobras possíveis a partir da fita inicial e checar se é possível chegar ao estado final.