# Maratona de Programação de 2012

## Instituto de Matemática e Estatística
## Universidade de São Paulo

# Explicação dos Problemas

**Departamento de Ciência da Computação IME-USP**

# Problema A: Festival das noites brancas

Autor do problema: Marcel K. de C. Silva
Análise: Guilherme Souza
História: Carlinhos

O problema pede para calcular os três últimos dígitos do $n$-ésimo número de Fibonacci. Isso é equivalente a calcular $f(n)$ módulo 1000. Vamos ver então como podemos calcular isso de maneira rápida, já que $n$ vai até cerca de $2^{10000}$.

Como:

$$\begin{gathered} f(n) = f(n-1) + f(n-2) \Rightarrow \\ \begin{bmatrix} f(n+1) \\ f(n) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} f(n) \\ f(n-1) \end{bmatrix} \Rightarrow \\ \begin{bmatrix} f(n+1) \\ f(n) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \left( \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} f(n-1) \\ f(n-2) \end{bmatrix} \right) = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^2 \begin{bmatrix} f(n-1) \\ f(n-2) \end{bmatrix} \end{gathered}$$

Repetindo o processo chegamos que:

$$\begin{bmatrix} f(n+1) \\ f(n) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n \begin{bmatrix} f(1) \\ f(0) \end{bmatrix}.$$

De onde podemos obter o valor desejado $f(n)$, assumindo que $f(0) = 0$.

Agora temos que calcular rapidamente $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$. Primeiro pré-calcularmos os valores de $A_0 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^1$, $A_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^2$, $A_2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^4$, ..., $A_{10000} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{2^{10000}}$. Tendo $n$ em sua representação binária, para calcular o valor de $A$ basta multiplicar as matrizes $A_i$ para valores de $i$ correspondentes aos bits acesos (1) de $n$. Por exemplo se $n = 11010_2 = 2^1 + 2^3 + 2^4 = 26$, calculamos $A_1 A_3 A_4$, uma vez que

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{26} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{2+8+16} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^2 + \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^8 + \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{16} = A_1 A_3 A_4.$$

Por fim podemos calcular as matrizes $A_i$ usando o fato que $A_{i+1} = A_i A_i$. Assim, a complexidade da solução fica $O(M)$, sendo $M$ o tamanho da entrada (número de dígitos binários), que no caso é no máximo 10000.

## Problema B: Pontes de São Petersburgo

Autor do problema: Renato Parente
Análise: Guilherme Souza
História: Carlinhos

A primeira vista, o problema parece ser relacionado com grafos, mas na verdade é o clássico problema da mochila (*Knapsack*) disfarçado. Já que arestas em comum são contadas duas vezes, só importa o grau de cada vértice.

Assim, o problema é equivalente ao seguinte: dado um conjunto de elementos com valores associados (o grau de cada vértice), dizer se existe algum subconjunto cuja soma seja igual a $K$, sem poder escolher o mesmo elemento mais de uma vez. Esse problema pode ser resolvido por programação dinâmica:

$\triangleright\ v[1..n]$ contém os graus de cada vértice
$\triangleright\ ok[0..K]$ guarda se a soma $i$ pode ser atingida, começa com 0
$ok[0] = 1$
para cada vértice $v[i]$ :
    para $j$ de $K$ até $v[i]$ : $\triangleright$ decresce para evitar repetição
        se $(ok[j - v[i]] == 1)$:
            $ok[j] = 1$ $\triangleright$ se a soma $j - v[i]$ é possível, então $j$ também é.

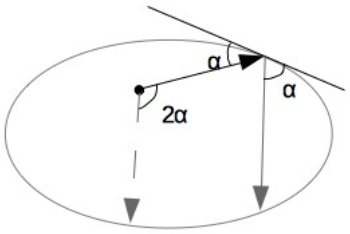
Assim basta verificar depois se $ok[K]$ é 1.

## Problema C: Myachowski, o futebol russo

Autor do problema: Cristiane M. Sato
Análise: Guilherme Souza
História: Carlinhos

O problema pode ser dividido em duas partes. A primeira é dado um ponto dentro da elipse e uma direção, achar a interseção da trajetória com a elipse. Para isso, uma solução seria achar a equação da reta e igualar a elipse, e tomar cuidado para pegar a solução certa (existe a solução indo no sentido oposto). Uma solução mais simples é fazer uma busca binária: conseguimos responder facilmente se um dado ponto está dentro ou não da elipse (basta comparar $\frac{x^2}{a^2} + \frac{y^2}{b^2}$ com 1). Então, sendo o ponto inicial $(p_x, p_y)$ e a direção $\vec{D} = (d_x, d_y)$, a trajetória é da forma $(p_x + md_x, p_y + md_y)$ e fazendo busca binária em $m$ conseguimos achar a interseção com a elipse com precisão arbitrária.

A segunda parte do problema é achar a nova trajetória após a reflexão. Para isso, um modo (entre vários) é achar o ângulo $\alpha$ da figura abaixo e rotacionar o vetor direção por $-2\alpha$ (no sentido antihorário).




Para rotacionar o vetor direção usamos a matriz de rotação

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Para calcular o ângulo $\alpha$ precisamos antes achar a tangente no ponto da interseção. A inclinação da tangente pode ser achada de vários modos, como achando a derivada da equação da elipse, resultando em $\vec{T} = (a^2 r_y, -b^2 r_x)$ sendo $(r_x, r_y)$ o ponto na elipse. Por fim, o ângulo $\alpha$ pode ser calculado como: $\arctan(\alpha) = \frac{\vec{T} \times \vec{D}}{\vec{T} \cdot \vec{D}}$ onde $\vec{T} \times \vec{D}$ é o produto vetorial (de módulo igual à $|\vec{T}||\vec{D}|\sin(\alpha)$) e $\vec{T} \cdot \vec{D}$ é o produto escalar (de módulo igual à $|\vec{T}||\vec{D}|\cos(\alpha)$).

Assim, após achar a primeira interseção, achamos o ângulo, rotacionamos a direção e achamos o segundo ponto.

## Problema D: Cerco a Leningrado

Autor do problema: Marcio T. I. Oshiro
Análise: Guilherme Souza
História: Carlinhos

Uma vez que o soldado tem que matar todos os atiradores do caminho e tem apenas uma chance para cada, o caminho pelo qual ele tem maior chance de chegar no destino é aquele no qual ele encontra o menor número de atiradores, de modo que o problema se reduz a achar o caminho de menor custo em um grafo, que pode ser resolvido pelo algoritmo de Dijkstra.

Após achar o caminho de menor custo $C$, ou seja, o caminho que passa pelo menor número de atiradores, é simples achar a probabilidade do soldado sobreviver. Se $C > K$ a probabilidade é 0, já que não tem munição suficiente para matar todos. Caso contrário, note que os eventos de acertar um determinado atirador são independentes. Logo, a resposta é $P^C$, pois para cada atirador do caminho ele tem probabilidade $P$ de acertar o primeiro tiro e não pode errar nenhum tiro.

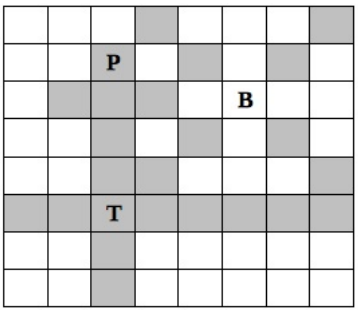
## Problema E: Desafio de São Petersburgo

Autor do problema: Marcio T. I. Oshiro
Análise: Guilherme Souza
História: Carlinhos

Esse problema pede para simular um estado em uma partida de xadrez e portanto não requer conhecimentos específicos, sendo a dificuldade achar uma maneira fácil e simples de implementar as peças, tomando cuidado com os movimentos de cada peça e o fato de que o ataque de cada peça não atravessa as outras peças do tabuleiro. Outro caso que tem que tomar cuidado é que o ataque das peças pretas a princípio não atravessa o rei branco, mas quando esse se mover pode ser que atravesse, de modo que é mais fácil desconsiderar o rei branco quando for analisar quais casas são atacadas.

Um modo de fazer isso é ter uma matriz correspondente ao tabuleiro, guardando a informação se tem uma peça preta na casa e outra matriz guardando se a casa é atacada por alguém. Então inicialmente se marca as casas onde tem peças pretas e depois pra cada peça preta se marca as casas que ela ataca, tomando o cuidado para não atravessar outra peça. Por fim, basta ver se a posição do rei branco é atacada e se os oito vizinhos dela também são (tomando cuidado com posições fora do tabuleiro), e lembrar que o rei branco pode comer alguma peça preta, desde que a casa em si não seja atacada por outra peça.

Por exemplo, no seguinte tabuleiro as seguintes posições ficariam atacadas:



## Problema F: Os benefícios da vodka

Autor do problema: Marcio T. I. Oshiro
Análise: Guilherme Souza
História: Carlinhos

Esse problema é uma aplicação clássica do problema de corte mínimo. Sejam $b(p_i)$ o quanto cada categoria $p_i$ dá de benefício e $c(q_i)$ quanto cada tipo $q_i$ de vokda custa. Para uma dada escolha de compra de tipos de vodka, seja $P$ o conjunto das categorias não escolhidas e $Q$ os do tipos escolhidos. Note que

$$\sum_{p_i \notin P} b(p_i) = \sum_i b(p_i) - \sum_{p_i \in P} b(p_i).$$

Assim, o benefício total é:

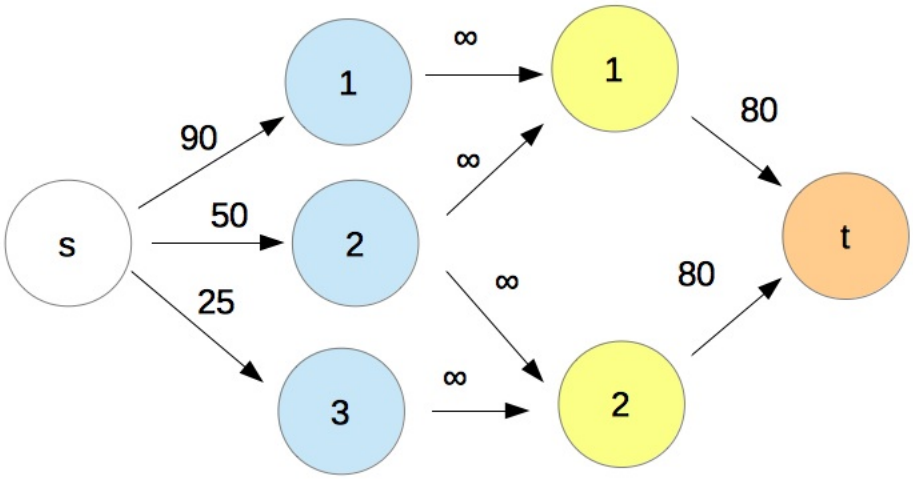
$$B = \sum_i b(p_i) - \sum_{p_i \in P} b(p_i) - \sum_{q_i \in Q} c(q_i).$$

Queremos então maximizar $B$. Mas, como $\sum_i b(p_i)$ é constante, queremos então minimizar

$$\sum_{p_i \in P} b(p_i) + \sum_{q_i \in Q} c(q_i).$$

Considere então o seguinte grafo. Ligue a uma fonte $s$ cada tipo de categoria com um arco de capacidade $b(p_i)$; cada categoria a seus tipos necessários com um arco de capacidade infinita; e por fim cada tipo se liga com um ralo com um arco de capacidade $c(q_i)$. Então o problema se torna achar o corte mínimo desse grafo, o que é equivalente a achar o fluxo máximo e pode ser reslvido por qualquer algoritmo de fluxo. No fim então as categorias escolhidas seriam aquelas cujas arestas ligadas a fonte não foram usadas no corte, e o resultado final é calculado pela primeira fórmula.

Um exmplo corresponde ao primeiro caso de exemplo seria:

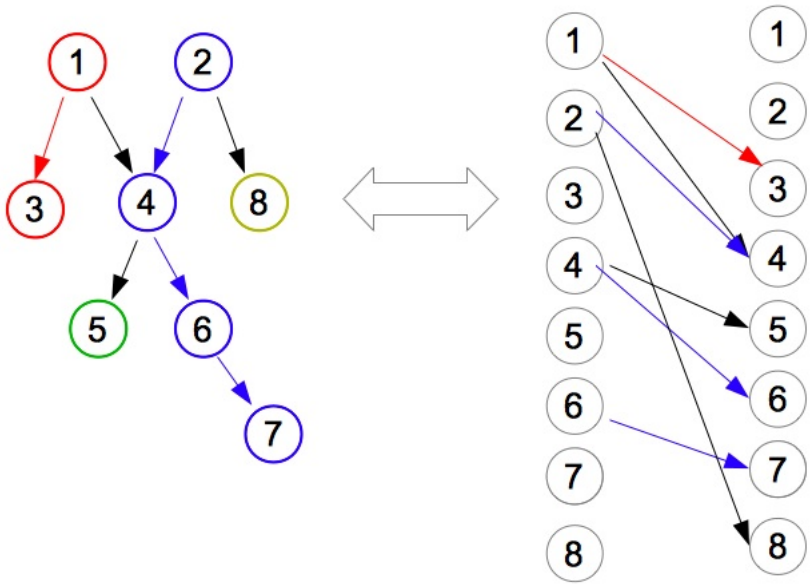



Onde o corte mínimo seria escolher os arcos de $s$ para 2 e 3 e o arco de 2 para $t$, resultando em um corte de 155. Logo, o lucro total é $90 + 50 + 25 - 155 = 10$.

## Problema G: As dinastias de São Petersburgo

Autor do problema: Gabriel R. da C. Peixoto
Análise: Guilherme Souza
História: Carlinhos

O problema pede para achar o menor número de caminhos (as dinastias) disjuntos que compõem uma dada "árvore" genealógica. Esse problema pode ser reduzido para achar um emparelhamento (*matching*) máximo em um grafo bipartido. Considere o seguinte grafo bipartido. Cada vértice aparece uma vez de cada lado e existe uma aresta entre dois vértices se e somente se existe a aresta na árvore original. Então cada caminho no grafo tem um caminho equivalente no matching, e vice versa. Por exemplo, para a seguinte árvore genealógica, o emparelhamento correspondente é:




Onde as arestas coloridas indicam o caminho e as arestas usadas no emparelhamento máximo. Portanto a resposta do número de caminhos é o número de vértices menos o emparelhamento máximo, e o problema é resolvido então por qualquer algoritmo de emparelhamento eficiente, já que o número de arestas e vértices é relativamente alto ( até 1000 vértices e 10000 arestas).

## Problema H: Festas de São Petersburgo

Autor do problema: Marcio T. I. Oshiro
Análise: Guilherme Souza / Marcio T. I. Oshiro
História: Carlinhos

O problema pede para escolher um grupo de pessoas no qual cada uma tenha pelo menos $K$ amigos entre si. Portanto, se uma pessoa tem inicialmente menos que $K$ amigos, podemos desconsiderar elas e para cada amigo dela reduzir em um o números de amigos. Com isso, mais pessoas com menos de $K$ amigos podem aparecer e analogamente podemos desconsiderá-las. Podemos fazer isso até que somente sobrem pessoas com pelo menos $K$ amigos, ou não sobre ninguém.

Existem três maneiras de fazer isso. Na primeira mantemos guardado para cada um quantos amigos ele ainda tem. Então buscamos se alguém tem menos que $K$ amigos, e se tiver para cada amigo dela reduzimos em 1 o número de amigos. A complexidade final é $O(N^2)$, pois cada pessoa pode ser retirada no máximo uma vez e a cada passo percorremos todo mundo para achar alguém com menos de $K$ amigos e se existir percorremos todos os amigos dela.

Uma outra solução é usar lista de adjacências para manter as relações de amizade e usar uma estrutura de dados como fila de prioridade (*priority queue*) (onde conseguimos achar o maior/menor elemento em $\log(N)$) que mantém quantos amigos cada pessoa tem, de modo que podemos obter em $\log(N)$ a pessoa com menos amigos, achando se tem alguém com $K$ ou menos amigos, e sendo o resto da solução análoga. Assim a complexidade final fica $O(N \log N + M)$, pois a cada passo gastamos $\log(N)$ para achar a pessoa com menos amigos e cada relação de amizade é percorrida uma vez.

Uma terceira solução é usar lista de adjacências para manter as relações de amizade e uma fila (ou pilha) para guardar quem tem menos do que $K$ amigos. Inicialmente a fila contém todos com menos de $K$ amigos e a cada iteração removemos uma pessoa da fila e diminuimos em um a quantidade de amigos de cada um de seus amigos. Caso algum desses amigos tenha ficado com menos do que $K$ amigos, então o colocamos na fila. Esse processo continua até que a fila fique fazia. No final, o conjunto de pessoas que nunca entraram na fila é o conjunto desejado. Neste caso, como as operações em filas são constantes, a complexidade fica $O(N + M)$.

Para os limites dados, ambas as soluções devem passar, já que $N$ é menor ou igual a 1000 e o grafo pode chegar a ser completo.

## Problema I: Produção ótima de ótima vodka

Autor do problema: Marcio T. I. Oshiro
Análise: Guilherme Souza
História: Carlinhos

Inicialmente vamos pensar em apenas achar o melhor custo pra depois achar uma solução. O importante do problema é quantos anos passaram e qual a idade da máquina atual. Assim, podemos resolver com uma DP[1] cujo estado é [quantos anos já foram][idade da máquina atual] que guarda a melhor solução possível. Cada estado depende de outros dois, dependendo se troca a máquina atual (o que tem que ser feito se a idade for $M$) ou a mantém, e conseguimos terminar a DP quando chegarmos em $N$ anos partindo do ano 0 com a máquina de idade $I$. Como cada estado é resolvido em O(1) e são O($NM$) estados, a complexidade total é O($NM$).

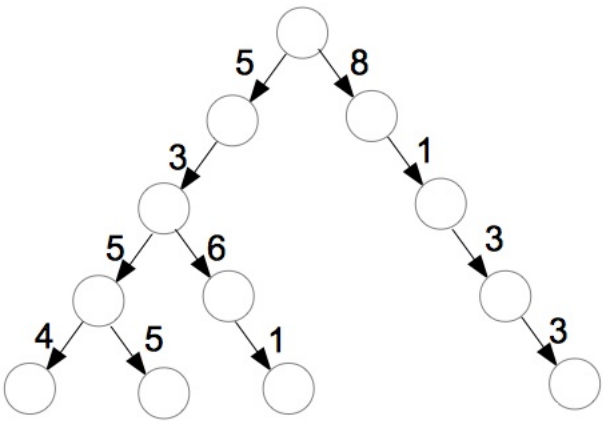
Para achar a sequência da solução, uma vez que queremos a solução trocando uma máquina sempre que der, guardamos também se o melhor possível para cada estado é atingido trocando a máquina ou não (se tanto faz guardamos que dá pra trocar). Agora partimos do estado inicial e vemos a cada passo se podemos trocar máquina ou não, e avançando o estado de acordo até chegar no ano $N$.

[1] Programação dinâmica (*dynamic programming*).

## Problema J: Lista telefônica econômica

Autor do problema: Renato Parente
Análise: Guilherme Souza
História: Marcio T. I. Oshiro

Um jeito de resolver é combinar trie com programação dinâmica. Uma trie é uma estrutura de dados no formato de árvore que armazena várias strings, sendo acada aresta um caractere de modo que cada caminho da arvore até a raiz é uma string dada. Por exemplo, para as strings 5354, 5355, 5361 e 8133 temos a seguinte trie:




E então obtemos o número de caracteres impressos contando o número de arestas da trie, de forma que o número de carcateres economizados é o total de caracteres menos os impressos. Isso ocorre já que a parte em comum dos telefones aparecem apenas uma vez como uma aresta. A complexidade dessa solução é $O(NS)$, onde $N$ é o número de telefones e $S$ o tamanho de cada um.

Outra solução possível é considerar os números ordenados e contar entre dois telefones consecutivos quantos caracteres são economizados (equivale a uma busca andando sempre para o caractere menor na trie). A complexidade fica um pouco maior ($O(NS \log N)$ para a ordenação + $O(NS)$ para o cálculo da resposta) mas ainda é rapida o suficiente para resover o problema.