# Maratona de Programação
# Semana de Informática 2023

André G. Santos[1] Salles V. G. Magalhães[1]

[1]Departamento de Informática
Universidade Federal de Viçosa (UFV), Brazil

Semana de Informática, 2023

# B. Questões ausentes  $(19/21)^1$

Resumo

- Questões em ordem alfabética, mas faltam as vogais
- Ler a letra da última questão e informar quantas questões existem

Solução 1

- Simulação: contar A até a letra da última, não contando vogais

Solução 2

- cont = letra - 'A' + 1 é quantidade se não faltasse letra
- para cada vogal x, se x $<$ letra, cont−=1
- pode até ser feito com uma conta apenas:

```
(letra-‘A’+1)-(letra>=‘A’)-(letra>=‘E’)-(letra>=‘I’)-(letra>=‘O’)-(letra>=‘U’)
```

[1] quantos times passaram a questão e número de submissões

# C. Números unários (19/22)

Resumo

- Número unário é uma sequência de 1's
- Ex.: $111 = 1 \times 1^2 + 1 \times 1^1 + 1^0 \times 1$
- Ler dois números unários e informar o resultado da subtração deles

Solução

- Note que o valor de um número unário é a quantidade de 1's
- Ler dois strings e calcular a diferença de tamanho

# L. Fome de pizza (19/21)

Resumo

- $N$ pizzas de raio $R_i$ com $P_i$ pedaços
- Determinar qual tem o maior pedaço (maior área)

Solução

- Tamanho do pedaço da pizza $i$: $T_i = \pi R_i^2 / P_i$
- Calcular cada uma e informar $i$ de maior $T_i$
- Note que não precisa usar $\pi$ para comparar dois pedaços:

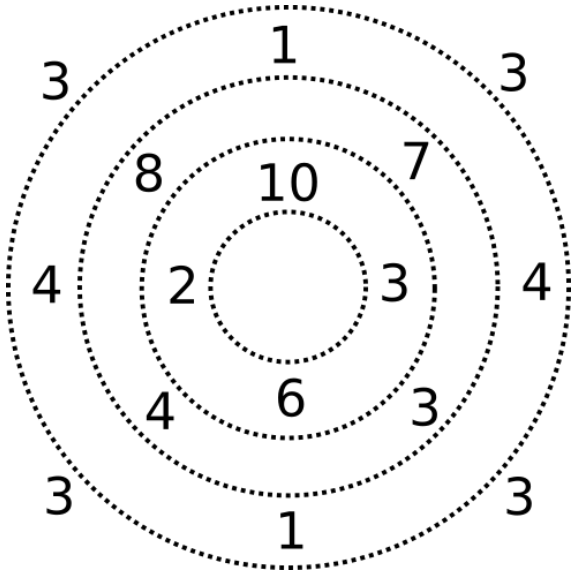
$$T_i > T_j \Rightarrow \quad \frac{\pi R_i^2}{P_i} > \frac{\pi R_j^2}{P_j} \Rightarrow \quad \frac{R_i^2}{P_i} > \frac{R_j^2}{P_j} \Rightarrow \quad R_i^2 P_j > R_j^2 P_i$$

- Eliminando $\pi$ e a divisão, evita problema de precisão de ponto flutuante

# J. Capivaras cochilando (15/32)

Resumo

- $N$ capivaras cochilando em círculo com $x_1$, $x_2$, ..., $x_n$ carrapatos
- Em volta delas, $N$ cochilando com $|x_1 - x_2|$, $|x_2 - x_3|$, ..., $|x_n - x_1|$
- E assim por diante, até círculo com todos $x_i$ iguais
- Informar qtd total de carrapatos, i.e., soma de $x_i$ de todos círculos



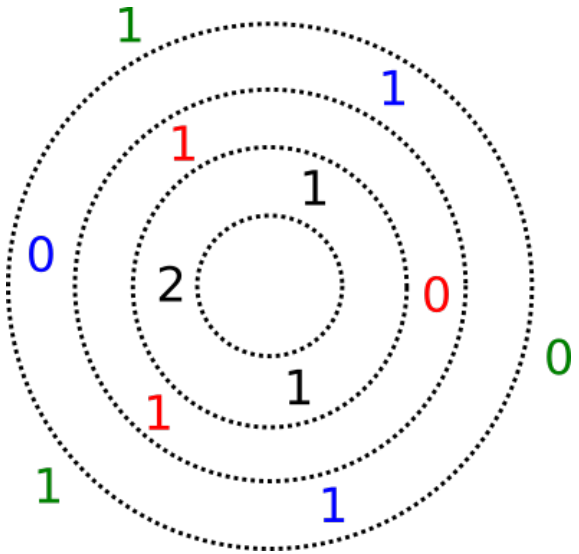

Solução

- Simulação: iterar gerando novos círculos até critério de parada
- Diferença entre vizinhos fica cada vez menor

# K. Capivaras dormindo (1/21)

Resumo

- Mesmo do anterior, mas círculo pode nunca terminar (imprimir -1)
- Exemplo: $2, 1, 1 \to \mathbf{1,0,1} \to 1, 1, 0 \to 0, 1, 1 \to \mathbf{1,0,1}$
- "Tamanho" do ciclo pode variar




Solução

- Teorema: se potência de 2, sempre termina (rápido!)
- Solução "correta": guardar sequências em um set e detectar ciclo – TLE
- Solução sem TLE: se fez muitas (ex: 10000) iterações, imprimir -1

# M. Filó e os primos gêmeos (14/26)

Resumo

- A capivara Filó fugiu e se escondeu entre os primos gêmeos
- Dado $N$, encontrar o primeiro $X > N$ com $X - 1$ e $X + 1$ primos

Solução

- Primos gêmeos são frequentes, basta iterar $P = N, N + 1, N + 2, \ldots$
- Se $P$ for primo e $P + 2$ também, escrever $P + 1$

# N. atan: onde está? (13/33)

Resumo

- Dado um texto, encontrar ocorrências de "atan"

Solução

- Mesmo soluções lentas passam (são apenas 1000 linhas de até 1000 colunas)

```
int main() {
    int linha = 0;
    string st;
    while(getline(cin,st)) {
        linha++;
        for(int c=0;c<st.size();c++) {
            if(st.substr(c,4)=="atan") {
                cout << linha << " " << c+1 << "\n";
            }
        }
    }
```

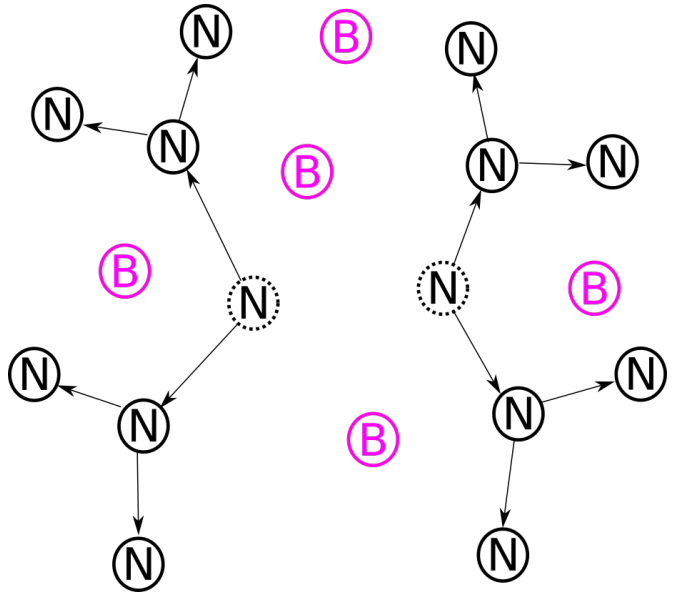
- Sem usar funções da classe string:

```
if (st[i]=='a' && st[i+1]=='t' && st[i+2]=='a' && st[i+3]=='n')
```

# F. Capyheimer (5/36)

Resumo

- Cada capivara que assiste ao filme do Capyheimer convence mais (até) 2 assistirem no dia seguinte (exceto as que gostam da Capybarbie)
- Dados o total de neutras, de fãs Capybarbie, e inicial
- Determinar o máximo de capivaras que assistirão ao filme em um dia




Solução

- Simulação simples
  - Exemplo: 14 neutras, $X$ Capybarbie, 2 inicial: $2 + 4 + 8 \rightarrow 8$
  - Cuidado : 9 neutras, $X$ Capybarbie, 2 inicial: $2 + 4 + 3 \rightarrow 4$ (máximo pode não ser no último dia)
- Cuidado com overflow!

# H. DJ Capialok  (5/6)

Resumo

- Capialok quer escolher as músicas que mais empolgam na formatura
- Duração $D_i$ de cada música de duração $T$ da festa
- Evento teste para calcular pontos (empolgação) de cada uma

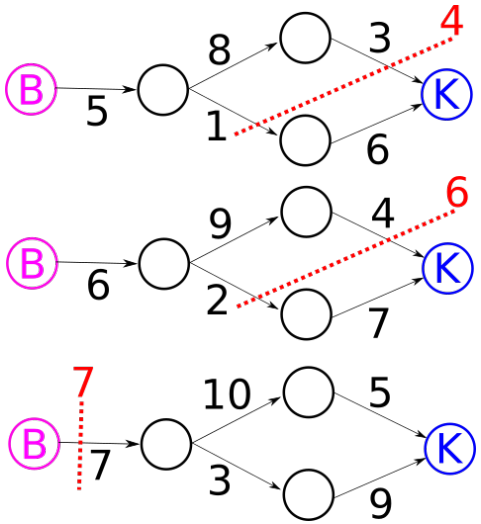
Solução

- Música 1 vale 100 pontos
- Para as demais:
  - Se for + e a anterior também, 1 a mais que a anterior
  - Se for − e a anterior também, 2 a menos que a anterior
  - Senão, vale 100
- Problema da mochila com valor = pontos e peso = duração

# G. Capybarbie (0/10)

Resumo

- Capybarbie quer viajar de 0 até $N - 1$
- Vilão com $B$ bombas. Cada estrada precisa de algumas bombas para ser destruída
- Se Capyken investir $X$, número de bombas para cada estrada subirá em $X$ unidades




Solução

- Corte mínimo em Grafo com Dinic: encontra quantidade de bombas para impedir tráfego
- Criar função que, dado valor investido, fala se haverá tráfego
- Se testar todos possíveis valores a investir, teremos algo do tipo: $FFFFTTTT \rightarrow$ Busca binária

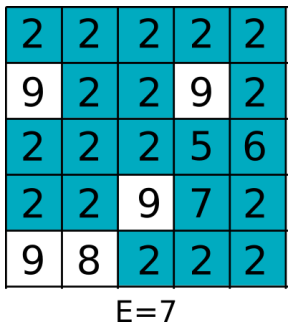
# P. Ilhas na UFV (0/1)

Resumo

- Matriz $M$ com $N \times N$ células. Cada uma tem uma elevação.
- $Q$ consultas do tipo: dada uma altura de água, quantas ilhas há em $M$ ?

Solução DFS/BFS $O(QN^2)$ – TLE

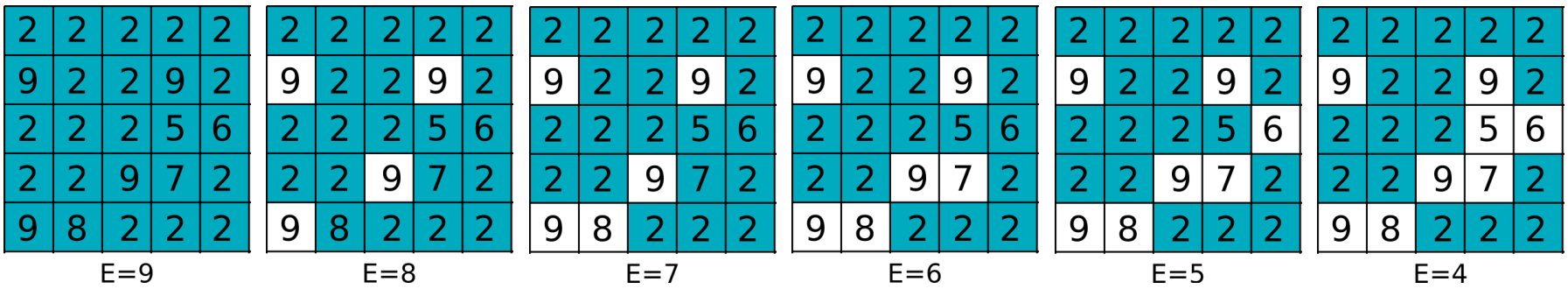
- Para cada consulta com elevação $E$, calcular componentes conexos em $M$ considerando que estão secas apenas as células com valor $> E$



# P. Ilhas na UFV (0/1)

Solução ordenação + Union-Find $O(N^2 logN)$

- Pré-calcular quantas ilhas há em cada elevação possível (todos números que aparecem na matriz)
- Ordenar células por elevação (decrescente)
- Ao processar célula: "ativar" na matriz e considerar um novo CC.
- Usar Union-Find para saber quais células ativas (secas) se juntaram (formando ilhas maiores)



# D. Febre Maculosa na UFV (-/-)

Resumo

- Capivara infectada fez a rota $(i, yc_i)$. Ex.: $(1,1), (2,3), (3,5), (4,1)$
- Várias rotas retas de estudantes: $y, x_{start}, x_{end}$
- Determinar quantas vezes o segmento de cada estudante intercepta a rota da capivara

Solução $O(N \log N)$

- Fazer um *sweep-line* pelos $y$ da rota da capivara
- Manter segmentos ativos em *bit-tree*
- Cada segmento representado pela sua coordenada $x$ inicial
- Após cada linha de varredura podemos responder as consultas que estão entre ela e a próxima linha