K - Strike

Problema

- **Input:** Informação de companhias, e voos
- **Output:** Melhor dia para greve e número de voos cancelados

Restrições

- $1 \leq N \leq 20$
- $1 < M \leq 10000$
- $1 < d < 30$
- $Today - FromDay \in \{0, 1\}$
- $FromHour < ToHour \vee ToDay \neq FromDay$

Classificação

- **Categorias:** programação simples
- **Dificuldade:** Fácil

K - Strike

Solução

- Solução fácilmente obtida em tempo linear
- Precisamos
  - Um array para contabilizar voos cancelados por cada dia
  - Conjunto para guardar companhias aéreas relevantes
  - Conjunto para guardar aeroportos relevantes
- Estratégia
  - Ler cada voo
  - Descartar voos não relevantes
  - Se relevante, somar:
    - Somar +1 dia, no dia partida
    - +1 dia no dia chegada se diferentes
    - **Atenção**: só se soma o segundo dia se o aeroporto de chegada tb for relevante
  - No final retornar o índice e o valor máximo dos dias

D - Weekday Calculator

Problema

- **Input:** Lista de dias
- **Output:** Dia da semana

Limites

$1 \leq N \leq 10^4$	Dates
$d \in \{1, \ldots, 31\}$	day
$1900 \leq y \leq 5000000$	year
$m \in \{$ `Jan`, `Feb`, `Mar`, `Apr`, `May`, `Jun`, `Jul`, `Aug`, `Sep`, `Oct`, `Nov`, `Dec` $\}$.	month

Classificação

- **Categorias:** Ad Hoc
- **Dificuldade:** Fácil

D - Weekday Calculator

Zeller's congruence

- Resposta é mod7, onde $365 = 1$.
- h, $0 = Sat$, $1 = Sun$.
- q is day
- m is month $3 = March$, $\ldots$, $14 = Feb$
- $K = year\%100$.
- $J = year/100$.

$$h = \left(q + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \lfloor K/4 \rfloor + \lfloor J/4 \rfloor - 2J \right) \mod 7$$

I - Sprinkling Sprinklers

Problema

- **Input:** Obstáculos rectangulares num terreno.
- **Output:** Quantas células não têm obstáculos?

Limites

- $1 \leq L \leq 1000000$
- $1 \leq W \leq 100$
- $0 \leq O \leq 20000$
- $0 \leq X1 \leq X2 < L$
- $0 \leq Y1 \leq Y2 < W$

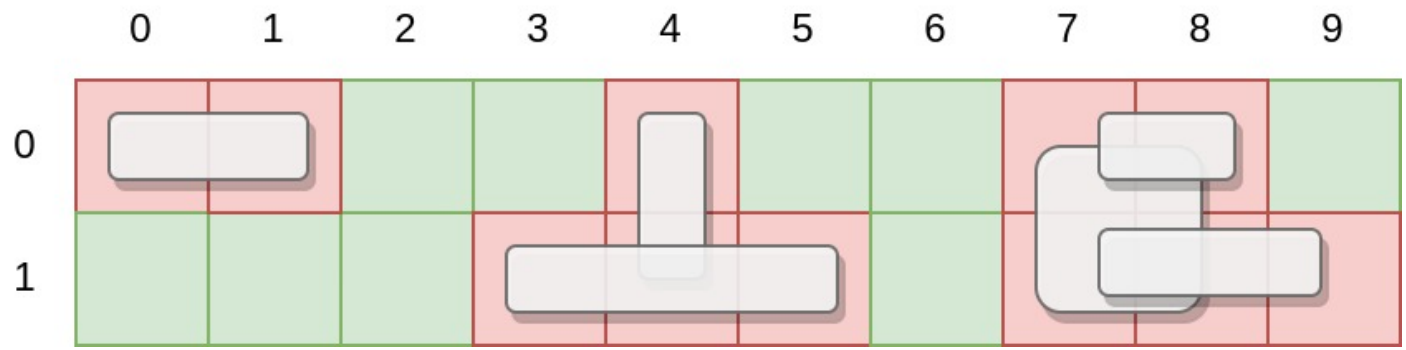
Classificação

- **Categorias:** sweep line, PD
- **Dificuldade:** Média

I - Sprinkling Sprinklers

Stacking

- George herda um terreno muito comprido mas estreito:
  - precisa de descobrir como irrigar todo o terreno.
  - ...colocando um *sprinkler* em cada célula livre.
  - mas o terreno tem muitos obstáculos.
  - ...e os obstáculos até podem estar sobrepostos.
- Numa imagem:

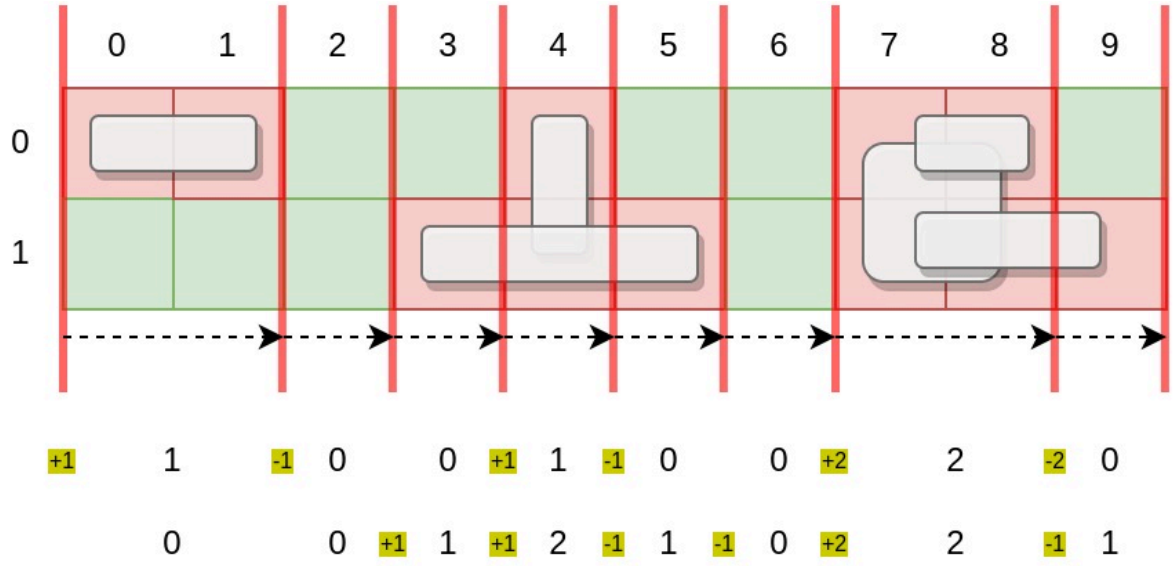


Q: Quantos *sprinklers* consegue colocar?

I - Sprinkling Sprinklers

Estratégia 1 : Sweep Line

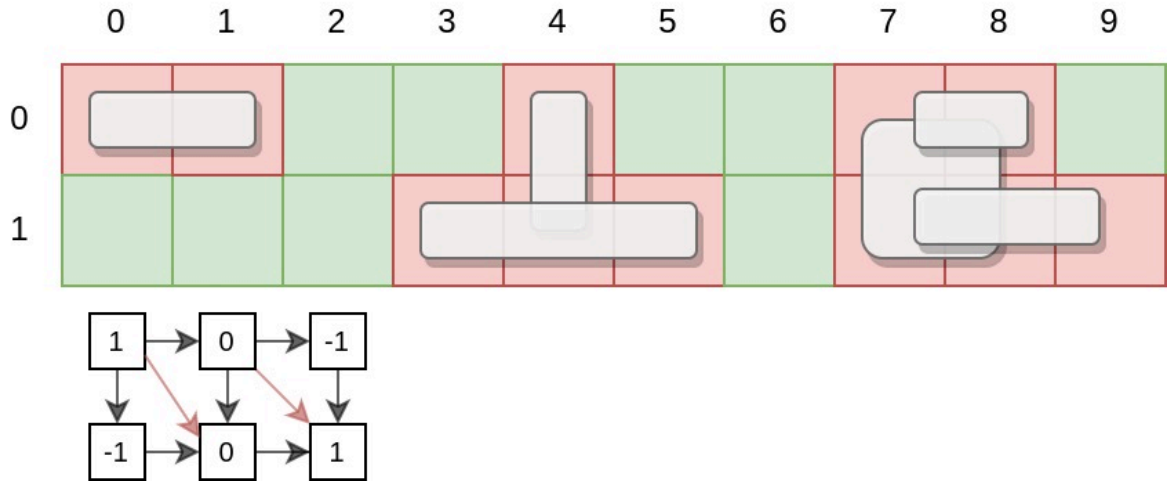
- Ordenar os obstáculos pelos X1s e pelos X2s (duas listas ordenadas).
- Percorrer pelas coordenadas ordenadas e ir somando quando um novo obstáculo entra ou subtraindo quando sai.
- Multiplicando pela distância entre as linhas.
- Pode ser feito uma linha de cada vez.



I - Sprinkling Sprinklers

Estratégia 2

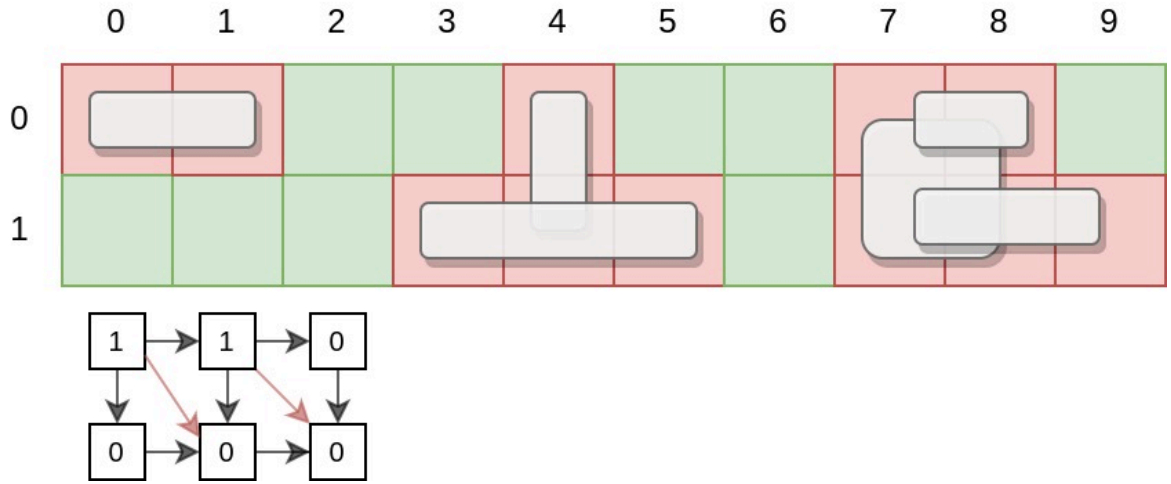
- Criar um array do tamanho do terreno.
- Percorrer os obstáculos e marcar no array quantos obstáculos entram e quantos saiem.
- Percorrer todo o terreno e contar as células vazias.
- Ir propagando os valores das células anteriores.



I - Sprinkling Sprinklers

Estratégia 2

- Criar um array do tamanho do terreno.
- Percorrer os obstáculos e marcar no array quantos obstáculos entram e quantos saiem.
- Percorrer todo o terreno e contar as células vazias.
- Ir propagando os valores das células anteriores.



E - Meeting with a friend

Problem

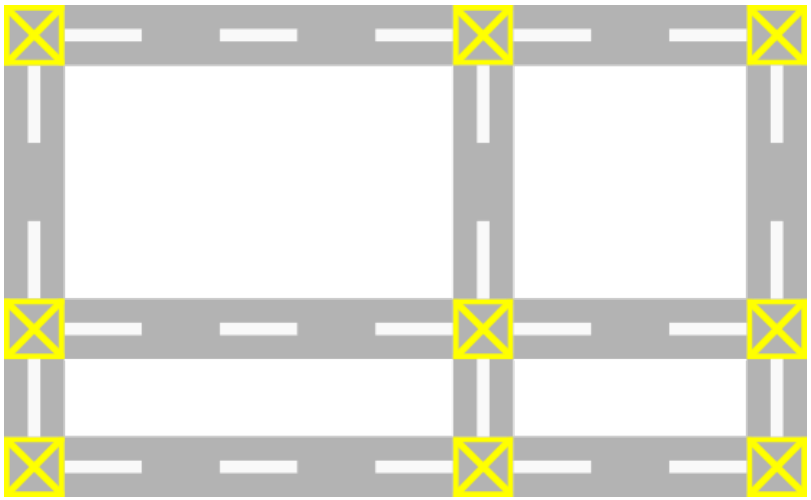
- **Input:** The coordinates of every intersection
- **Output:** The (lexicographically smallest) intersection where the absolute difference of the distances travelled by Ana and Pedro is minimal

Limits

- $1 \leq T \leq 100$
- $1 \leq N, M \leq 2 \times 10^5$
- $1 \leq X_i, Y_i \leq 10^8$

Classification

- **Categories:** binary search, two pointers
- **Dificulty:** Medium



E - Meeting with a friend

Example

```
1                 # T
4 3               # N M
1 7 10 14         # X_1, X_2, ..., X_N
1 4 6             # Y_1, Y_2, ..., Y_M
```

Ana is at $(1, 1)$, and Pedro is at $(14, 6)$.

If we consider intersection $(10, 4)$:

- Ana is at distance $(10 - 1) + (4 - 1) = 12$
- Pedro is at distance $(14 - 10) + (6 - 4) = 6$

Absolute difference is $|12 - 6| = 6$.

E - Meeting with a friend

Example

```
1                 # T
4 3               # N M
1 7 10 14         # X_1, X_2, ..., X_N
1 4 6             # Y_1, Y_2, ..., Y_M
```

Intersections $(7, 4)$ or $(10, 1)$, they are both at distance 9 from both And and Pedro, so the absolute difference is 0.

Other intersections have a larger absolute difference.

Then we should output $(7, 4)$ since it is lexicographically smaller.

E - Meeting with a friend

Solution 1

best $= X_N - X_1 + Y_M - Y_1$
$x = X_1$
$y = Y_1$
for $i = 1$ **to** N **do**
  for $j = 1$ **to** M **do**
    aux1 $= (X_i - X_1) + (Y_j - Y_1)$
    aux2 $= (X_N - X_i) + (Y_M - Y_j)$
    aux = abs(aux1 - aux2)
    if aux $<$ best **then**
      best = aux
      $x = X_i$
      $y = X_j$
    end
  end
end
return (x, y)

Complexity: $\mathcal{O}(N \times M)$ will give Time Limit Exceeded

E - Meeting with a friend

Solution 2

Note that for a fixed i and increasing $j = 1, 2, \ldots, M$, the absolute difference will first decrease towards zero, and then increase away from zero.

Then, taking into account the value of X_i, we can replace the second **for** with a binary search that searchs for the value in Y_j that would give the absolute difference closest to 0.

Complexity: $\mathcal{O}(N \log M)$ will give Accepted

E - Meeting with a friend

Solution 2

best/dist $= X_N - X_1 + Y_M - Y_1$
$x = X_1$
$y = Y_1$
for $i = 1$ **to** N **do**
  $v = \lfloor \text{dist}/2 \rfloor - X_i$
  /* Finds smallest index j such that $Y_j \geq v$ */
  $j = \text{lower\_bound\_index}(Y, v)$
  if $j \leq M$ **then**
    /* Check if X_i, Y_j gives a better absolute difference and if so update best, x, y */
  end
  if $j > 1$ **then**
    $j = j - 1$
    /* Check if X_i, Y_j gives a better absolute difference and if so update best, x, y */
  end
end
print(x, y)

G — Doubled Secrets

Problema

- **Input:** Mensagem cifrada, de comprimento L, constituída somente por letras minúsculas
- **Output:** Mensagem original, não vazia

Limites

$$2 \leq L \leq 20\,000$$

Classificação

Categorias: Algoritmos e estruturas de dados
Dificuldade: Fácil **Autor:** Vasco Pedro

G — Doubled Secrets

Solução

- Identificar par de letras repetidas mais afastadas

`wtreqbhcsatziwkkyniqsgdxgjnitsrxjyefta`

- Retirar tudo até à primeira ocorrência e a partir da última

`reqbhcsatziwkkyniqsgdxgjnitsrxjyef`

- Repetir enquanto houver alguma repetição

`reqbhcsatziwkkyniqsgdxgjnitsrxjyef`

G — Doubled Secrets

O tempo

Para ser eficiente

- Manter listas das ocorrências de cada letra

a: 10, 38 ... t: 2, 11, 29, 37 ...

- Procurar aquela em que a diferença entre o primeiro e o último elementos é maior

  Retirar das listas os elementos com posições que já não interessam

a: 10 ... t: 11, 29 ...

G — Doubled Secrets

O tempo

Para ser eficiente

- Manter listas das ocorrências de cada letra

a: 10, 38 ... t: 2, 11, 29, 37 ...

- Procurar aquela em que a diferença entre o primeiro e o último elementos é maior

  Retirar das listas os elementos com posições que já não interessam

a: 10 ... t: 11, 29 ...

Melhor...

- Manter as listas ordenadas por diferença entre os extremos

No Country for PhD and Engineers – 1

Problema

- A empresa quer contratar o *maior número* de doutores e engenheiros mas sem que o número de seguidores exceda a capacidade de produção

Classificação

- Problema da mochila

Restrições

$1 \leq C \leq 2^{31}$	capacidade de produção
$1 \leq N \leq 20$	número de candidatos
$1 \leq \text{len}\, m_a \leq 10$	nome candidato
$0 \leq f_a \leq 2^{20}$	número de seguidores
$1 \leq s_a \leq 1000$	pontuação CV

No Country for PhD and Engineers – 2

Solução

- Formulação como programação linear

$$\begin{aligned} \text{maximizar} \quad & \sum_{a=1}^{N} x_a \\ \text{sujeito a} \quad & \sum_{a=1}^{N} x_a f_a \leq C \\ \text{e} \quad & x_a \in \{0,1\} \qquad 1 - \text{contrata} \end{aligned}$$

- em caso de empate entre $\mathbf{x}^i$ e $\mathbf{x}^j = x_1 x_2 \ldots x_N$ escolhe o maior entre $\sum_{a=1}^{N} x_a^i s_a$ e $\sum_{a=1}^{N} x_a^j s_a$
  - em caso de empate, escolhe o vetor $\mathbf{x}$ lexicograficamente maior
- Gerar todas as combinações $\mathbf{x} = x_1 x_2 \ldots x_N$
- Devolver o melhor $\mathbf{x}$

No Country for PhD and Engineers – 3

Solução

$$f(i,h,\mathbf{x},f^S,s^S) = \begin{cases} (h,s^S,\mathbf{x}) & \text{se } i = N \\ f(i+1,h,\mathbf{x}0,f^S,s^S) & \text{se } f^S + f_i > C \\ \max(f(i+1,h,\mathbf{x}0,f^S,s^S), & \\ f(i+1,h+1,\mathbf{x}1,f^S+f_i,s^S+s_i)) & \text{caso contrário} \end{cases}$$

1. Caso paragem, não há mais candidatos, devolve tuplo com
   - número de contratados
   - soma do pontuação do CV
   - máscara bits
2. contratar o candidato i excede a capacidade de produção
3. devolve a melhor opção entre contratar ou não o candidato i

Invocar com $f(0,0,\lambda,0,0)$

No Country for PhD and Engineers – 4

```python
class Applicant:
    def __init__ (self):
        name, _followers, _score = input ().split ()
        self.m = name
        self.f = int (_followers)
        self.s = int (_score)


def loop (index, _number_hired, _sum_followers, _sum_score, _is_hired):
    if index == N:
        return _number_hired, _sum_score, _is_hired, _sum_followers
    elif As [index].f + _sum_followers > C:
        return loop (index + 1, _number_hired, _sum_followers, _sum_score, _is_hired + [False])
    else:
        sol1 = loop (index + 1, _number_hired, _sum_followers, _sum_score, _is_hired + [False])
        sol2 = loop (index + 1, _number_hired + 1, _sum_followers + As [index].f, _sum_score + As [index].s, _is_hired + [True])
        return max (sol1, sol2)


C = int (input ())
N = int (input ())
As =  [Applicant () for _ in range (N)]
number_hired, sum_score, is_hired, sum_followers = loop (0, 0, 0, 0, [])
print ('{} {} {}'.format (number_hired, sum_followers, sum_score))
for i, b in enumerate (is_hired):
    if b:
        print (As [i].m)
```

F — Was it a Dream?

Problema

- **Input:** Mapa $R \times C$, com obstáculo e saída, e T pares de coordenadas
- **Output:** Para cada par de coordenadas, o número mínimo de movimentos até à saída

Limites

$$1 \leq R, C \leq 900 \qquad 1 \leq T \leq 10$$

Classificação

Categorias: Grafos, BFS
Dificuldade: Médio– **Autor:** Vasco Pedro

F — Was it a Dream?

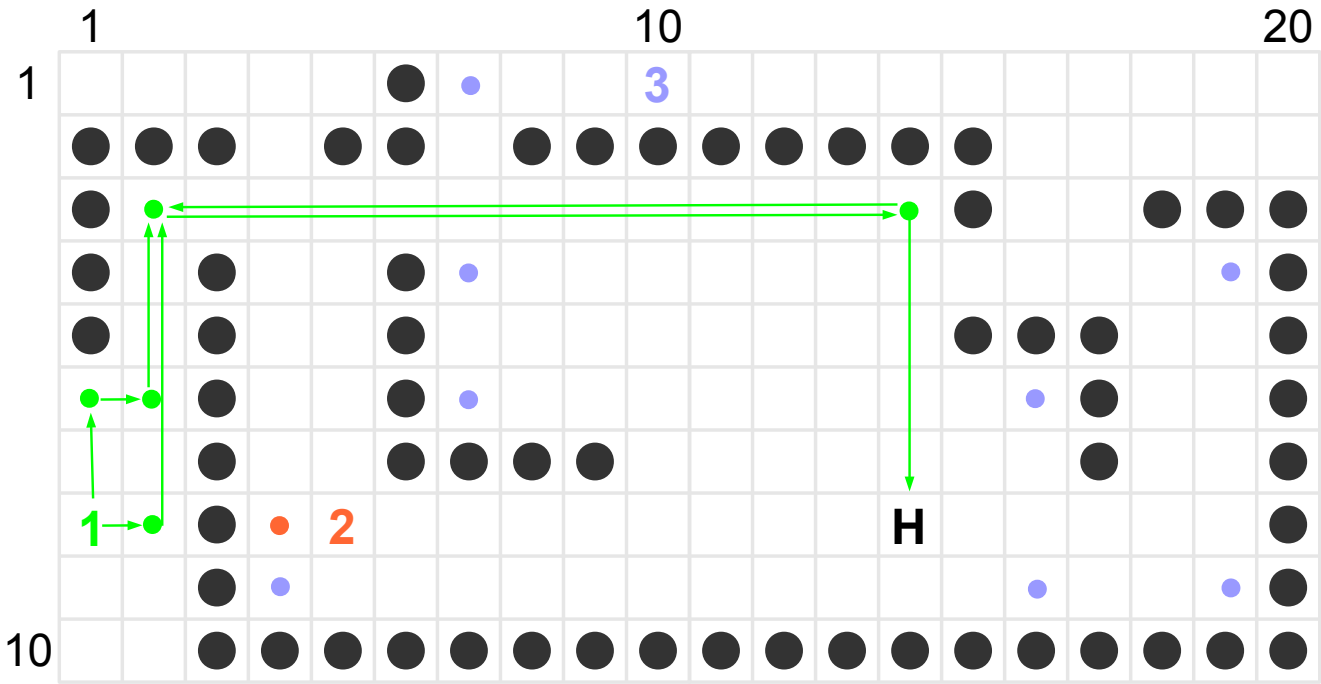
Solução

- Percurso em largura
- Grafo orientado
- Construção dinâmica do grafo

F — Was it a Dream?

Solução

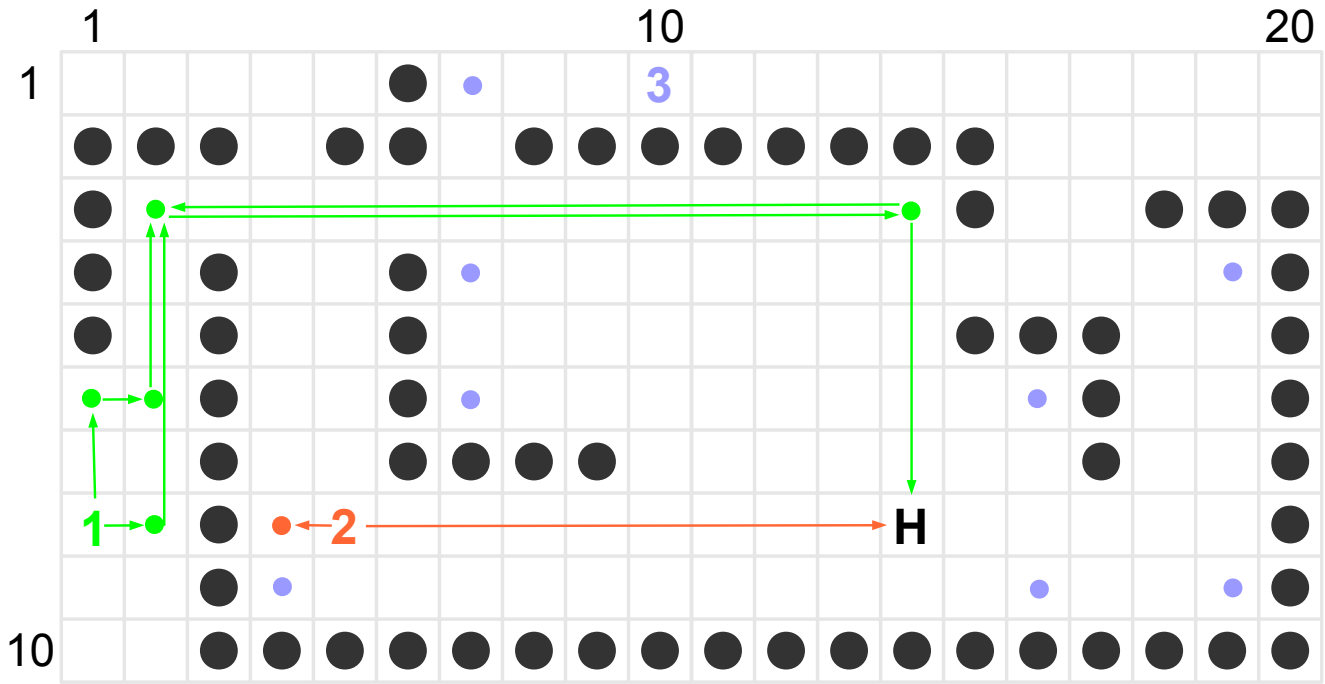
- Percurso em largura
- Grafo orientado
- Construção dinâmica do grafo (ou não. . .)



F — Was it a Dream?

Solução

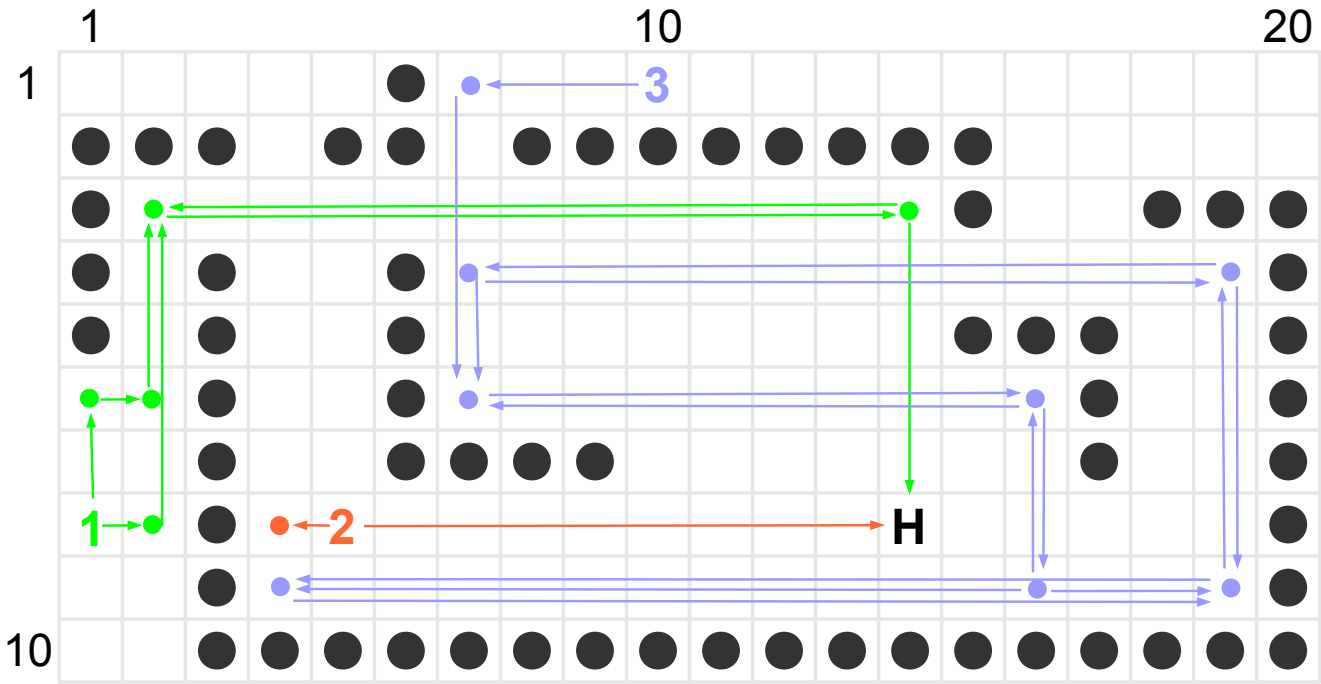
- Percurso em largura
- Grafo orientado
- Construção dinâmica do grafo (ou não. . .)



F — Was it a Dream?

Solução

- Percurso em largura
- Grafo orientado
- Construção dinâmica do grafo (ou não. . .)



J - Climate Change

Problema

- **Input:** n números inteiros t_i e n operações de atualizar um número ou fazer uma pergunta sobre um intervalo de posições $[a, b]$
- **Output:** Para cada intervalo de posições $[a, b]$ e um inteiro k indicar, quantos números dentro do intervalo são maiores que k

J - Climate Change

- $1 \leq n \leq 75\,000$
- $1 \leq m \leq 75\,000$
- $-10^8 \leq t_i \leq 10^8$

Classificação

- **Categorias:** Estruturas de Dados
- **Dificuldade:** Médio+/Difícil-

J - Climate Change

- **Exemplo:** quantas temperaturas acima de $k = 30$ no intervalo $[1, 9]$?

Dia	1	2	3	4	5	6	7	8	9	10
Temperatura	32	31	30	25	27	28	35	32	32	31

J - Climate Change

- **Exemplo:** quantas temperaturas acima de $k = 30$ no intervalo $[1, 9]$?

Dia	1	2	3	4	5	6	7	8	9	10
Temperatura	32	31	30	25	27	28	35	32	32	31

- **Solução bruta:** guardar valores num simples array
  - Update em $\mathcal{O}(1)$: alterar valor num array
  - Update em $\mathcal{O}(n)$: percorrer array de posição a até b

J - Climate Change

- **Exemplo:** quantas temperaturas acima de $k = 30$ no intervalo $[1, 9]$?

Dia	1	2	3	4	5	6	7	8	9	10
Temperatura	32	31	30	25	27	28	35	32	32	31

- **Solução bruta:** guardar valores num simples array
  - Update em $\mathcal{O}(1)$: alterar valor num array
  - Update em $\mathcal{O}(n)$: percorrer array de posição a até b
    Time Limit Exceeded

J - Climate Change

- **Exemplo:** quantas temperaturas acima de $k = 30$ no intervalo $[1, 9]$?

Dia	1	2	3	4	5	6	7	8	9	10
Temperatura	32	31	30	25	27	28	35	32	32	31

- **Solução bruta:** guardar valores num simples array
  - Update em $\mathcal{O}(1)$: alterar valor num array
  - Update em $\mathcal{O}(n)$: percorrer array de posição a até b
    Time Limit Exceeded

- Precisamos de maneira melhor de **"organizar"** os dados:
  - **Segment Tree:** árvore hierárquica de intervalos (em cada nível intervalos de metade do tamanho) - update de valor único em $\mathcal{O}(\log n)$
  - **Fenwick Tree:** binary indexed tree: partição em intervalos baseada nos bits (update de valor único em $\mathcal{O}(\log n)$)
  - **Sqrt Decomposition:** decomposição em intervalos de tamanho $\sqrt{(N)}$ (update de valor único em $\mathcal{O}(\sqrt{n})$)

J - Climate Change

- Como responder para **uma partição** que representa um intervalo?

J - Climate Change

- Como responder para **uma partição** que representa um intervalo?
  - Guardar um *"ordered multiset"* que permita responder **para um dado** k **qual é a posição dele**? (é o i-ésimo maior)

J - Climate Change

- Como responder para **uma partição** que representa um intervalo?
  - Guardar um *"ordered multiset"* que permita responder **para um dado** k **qual é a posição dele**? (é o i-ésimo maior)
  - Sabendo isso, sabemos quantos são maiores que k

J - Climate Change

- Como responder para **uma partição** que representa um intervalo?
  - Guardar um *"ordered multiset"* que permita responder **para um dado** k **qual é a posição dele**? (é o i-ésimo maior)
  - Sabendo isso, sabemos quantos são maiores que k
  - Como fazer isso? **extender uma árvore binária de pesquisa** (ex: quantos elementos debaixo de um nó?)
    - **policy based data structures** - **PBDS** (existe no GCC)
    - **Implementação manual** (ex: *treaps*)
    - Resposta para uma partição em $\mathcal{O}(\log n)$

J - Climate Change

- Como responder para **uma partição** que representa um intervalo?
  - Guardar um *"ordered multiset"* que permita responder **para um dado** k **qual é a posição dele**? (é o i-ésimo maior)
  - Sabendo isso, sabemos quantos são maiores que k
  - Como fazer isso? **extender uma árvore binária de pesquisa** (ex: quantos elementos debaixo de um nó?)
    - **policy based data structures** - **PBDS** (existe no GCC)
    - **Implementação manual** (ex: *treaps*)
    - Resposta para uma partição em $\mathcal{O}(\log n)$
  - Responder para várias partições "juntando" as várias respostas:
    - **Segment Tree:** $\mathcal{O}(\log n)$
    - **Fenwick Tree:** $\mathcal{O}(\log n)$
    - **Sqrt Decomposition:** $\mathcal{O}(\sqrt{n})$

J - Climate Change

- Como responder para **uma partição** que representa um intervalo?
  - Guardar um *"ordered multiset"* que permita responder **para um dado k qual é a posição dele**? (é o i-ésimo maior)
  - Sabendo isso, sabemos quantos são maiores que k
  - Como fazer isso? **extender uma árvore binária de pesquisa** (ex: quantos elementos debaixo de um nó?)
    - **policy based data structures** - **PBDS** (existe no GCC)
    - **Implementação manual** (ex: *treaps*)
    - Resposta para uma partição em $\mathcal{O}(\log n)$
  - Responder para várias partições "juntando" as várias respostas:
    - **Segment Tree:** $\mathcal{O}(\log n)$
    - **Fenwick Tree:** $\mathcal{O}(\log n)$
    - **Sqrt Decomposition:** $\mathcal{O}(\sqrt{n})$
  - Tempo total para m perguntas:
    - **Segment Tree+PBDS/Treaps:** $\mathcal{O}(m \times \log n \times \log n)$
    - **Fenwick Tree+PBDS/Treaps:** $\mathcal{O}(m \times \log n \times \log n)$
    - **Sqrt Decomposition+PBDS/Treaps:** $\mathcal{O}(m \times \sqrt{n} \times \log n)$

C - Walk Like an Archbishop or a Chancellor or a Queen

Problema

- **Input:** Número de casos de teste. Para cada teste, número de casas a visitar e casa de partida. Segue-se a lista de casas a visitar.
- **Output:** The minimum number of moves and the pieces that can finish in that number of moves.

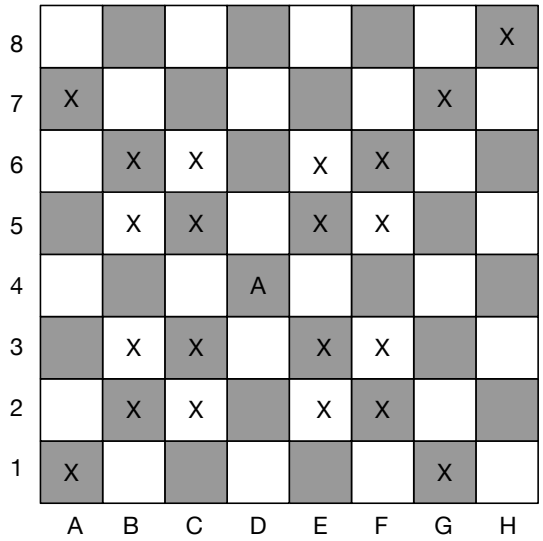
Limites

- $1 \leq n \leq 5$
- $1 \leq s \leq 10$

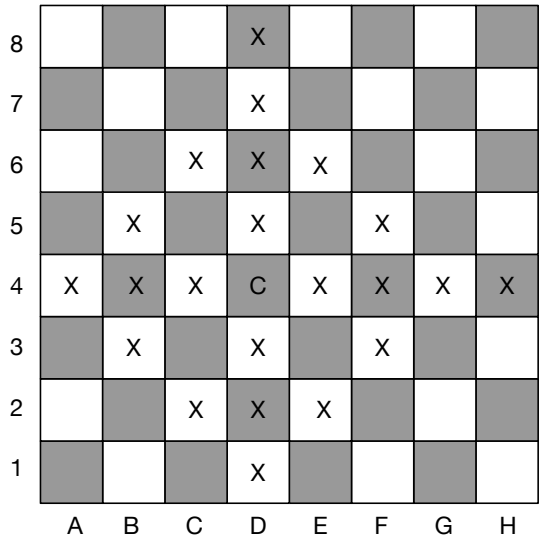
Classificação

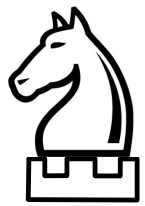
- **Categorias:** Grafos
- **Dificuldade:** Médio+

C - Walk Like an Archbishop or a Chancellor or a Queen

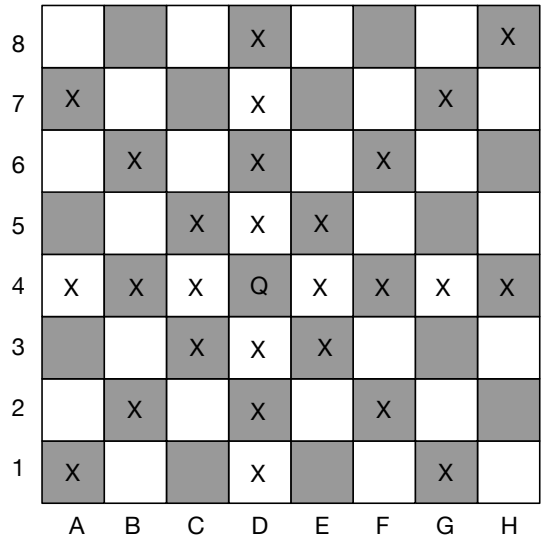


Archbishop





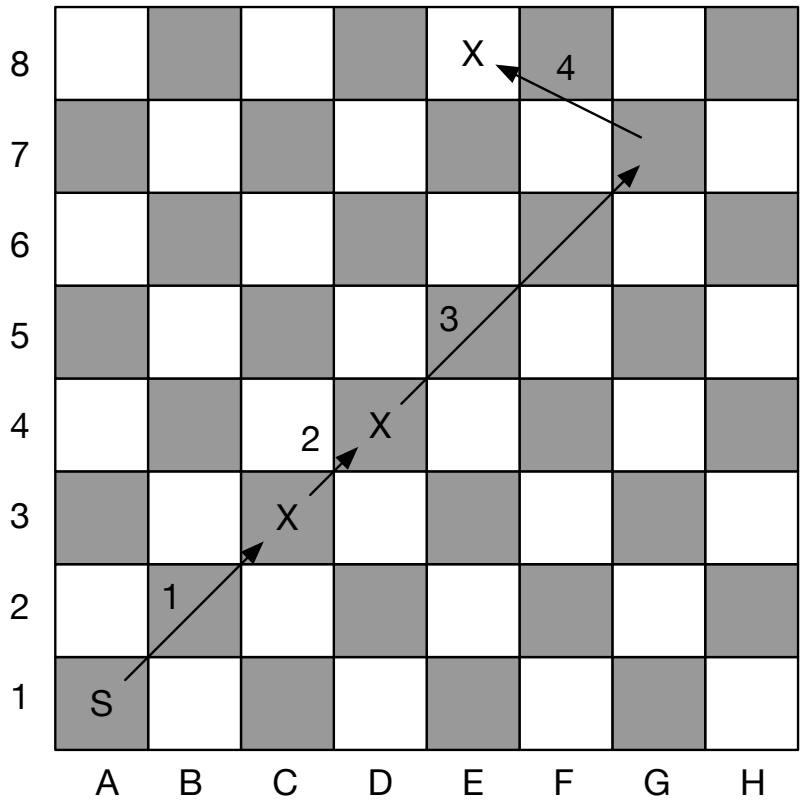
Chancellor



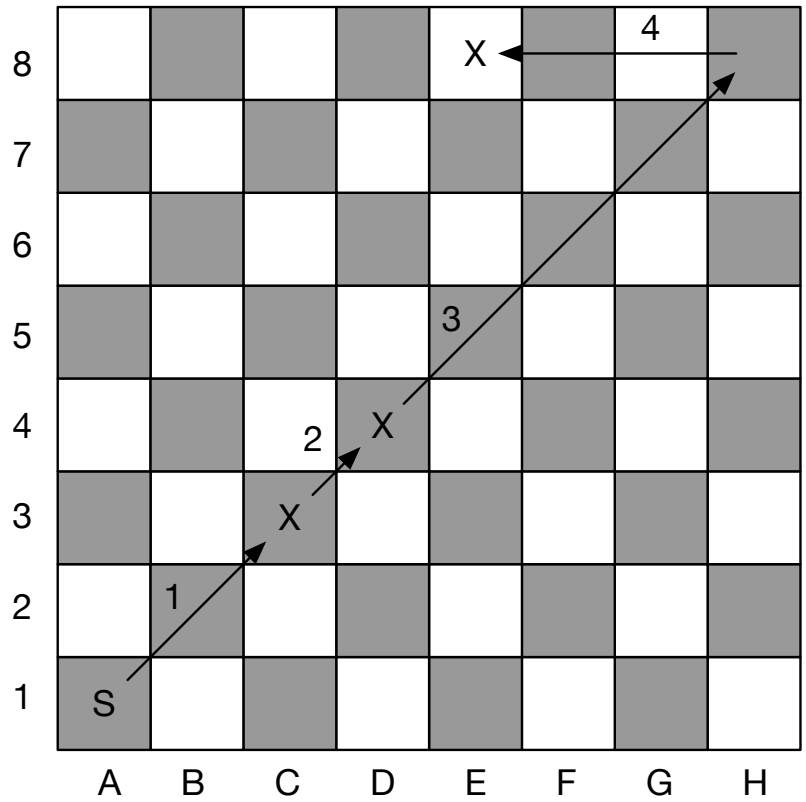
Queen

Pieces taken and adapted from vecteezy.com

C - Walk Like an Archbishop or a Chancellor or a Queen



Archbishop



Queen

Q: Quantos saltos para percorrer o percurso (e com que peças)?

C - Walk Like an Archbishop or a Chancellor or a Queen

Como resolver?

1. BFS para encontrar as distâncias entre todos os pares de casas a visitar (incluindo a de partida)
2. Usar força bruta permutando todas as ordens possíveis

B - Rescue by Rail

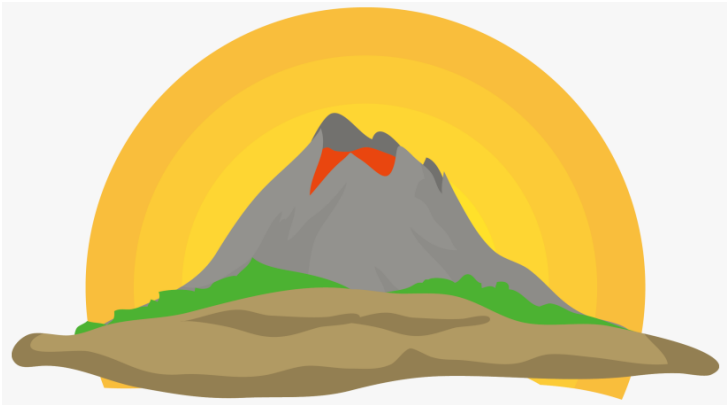
Problema

- **Input:** R regiões (uma região segura S) e L ligações entre regiões. Para cada região r: p_r habitantes e capacidade de evacuação d_r.
- **Output:** No máximo, quantas pessoas conseguem chegar a S?

Limites

$2 \leq R \leq 1\,000$ $\quad$ $1 \leq L \leq 5\,000$

$1 \leq p_r \leq 10\,000$ $\quad$ $1 \leq d_r \leq 300\,000$



Classificação

- **Categorias:** Grafos, Fluxo Máximo
- **Dificuldade:** Médio+
- **Autor:** Margarida Mamede

B - Rescue by Rail

- **Solução**
  - Construir uma rede de fluxos (V, E). Quais são os vértices, os arcos, as capacidades dos arcos, a fonte (*source*) e o dreno (*sink*)?

B - Rescue by Rail

- **Solução**
  - Construir uma rede de fluxos (V, E). Quais são os vértices, os arcos, as capacidades dos arcos, a fonte (*source*) e o dreno (*sink*)?
  - Calcular (o valor de) um fluxo máximo entre a fonte e o dreno.

B - Rescue by Rail

- **Solução**
  - Construir uma rede de fluxos (V, E). Quais são os vértices, os arcos, as capacidades dos arcos, a fonte (*source*) e o dreno (*sink*)?
  - Calcular (o valor de) um fluxo máximo entre a fonte e o dreno.
- **Construção do grafo**

B - Rescue by Rail

- **Solução**
  - Construir uma rede de fluxos (V, E). Quais são os vértices, os arcos, as capacidades dos arcos, a fonte (*source*) e o dreno (*sink*)?
  - Calcular (o valor de) um fluxo máximo entre a fonte e o dreno.
- **Construção do grafo**
  - 2 vértices por região x: x_1 (chegada) e x_2 (partida)

B - Rescue by Rail

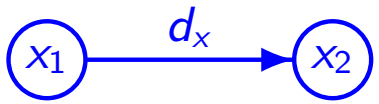
- **Solução**
  - Construir uma rede de fluxos (V, E). Quais são os vértices, os arcos, as capacidades dos arcos, a fonte (*source*) e o dreno (*sink*)?
  - Calcular (o valor de) um fluxo máximo entre a fonte e o dreno.
- **Construção do grafo**
  - 2 vértices por região x: x_1 (chegada) e x_2 (partida)
  - 1 vértice especial f (fonte)

B - Rescue by Rail

- **Solução**
  - Construir uma rede de fluxos (V, E). Quais são os vértices, os arcos, as capacidades dos arcos, a fonte (*source*) e o dreno (*sink*)?
  - Calcular (o valor de) um fluxo máximo entre a fonte e o dreno.
- **Construção do grafo**
  - 2 vértices por região x: x_1 (chegada) e x_2 (partida)
  - 1 vértice especial f (fonte)
  - 2 arcos por região x:

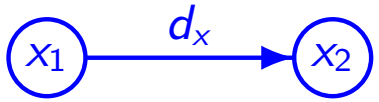
B - Rescue by Rail

- **Solução**
  - Construir uma rede de fluxos (V, E). Quais são os vértices, os arcos, as capacidades dos arcos, a fonte (*source*) e o dreno (*sink*)?
  - Calcular (o valor de) um fluxo máximo entre a fonte e o dreno.
- **Construção do grafo**
  - 2 vértices por região x: x_1 (chegada) e x_2 (partida)
  - 1 vértice especial f (fonte)
  - 2 arcos por região x:
    - (x_1, x_2) para que o número de pessoas que partem de x não exceda d_x

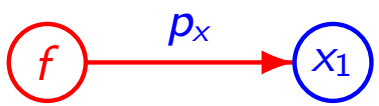


B - Rescue by Rail

- **Solução**
  - Construir uma rede de fluxos (V, E). Quais são os vértices, os arcos, as capacidades dos arcos, a fonte (*source*) e o dreno (*sink*)?
  - Calcular (o valor de) um fluxo máximo entre a fonte e o dreno.
- **Construção do grafo**
  - 2 vértices por região x: x_1 (chegada) e x_2 (partida)
  - 1 vértice especial f (fonte)
  - 2 arcos por região x:
    - (x_1, x_2) para que o número de pessoas que partem de x não exceda d_x

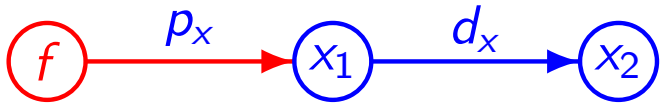


    - (f, x_1) para colocar a população de x (p_x) na região



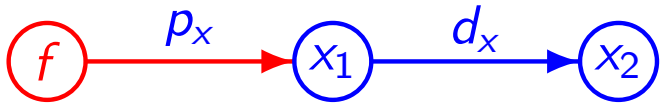
B - Rescue by Rail

- **Construção do grafo (cont.)**
  - 2 arcos por região x:



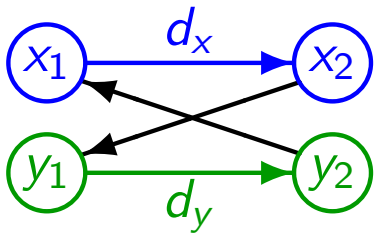
B - Rescue by Rail

- **Construção do grafo (cont.)**
  - 2 arcos por região x:



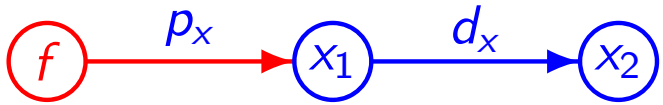
  - 2 arcos por cada ligação direta entre regiões x e y:

    (x_2, y_1) e (y_2, x_1), ambos com capacidade $+\infty$



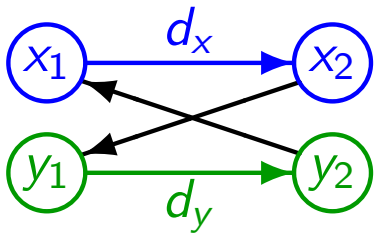
B - Rescue by Rail

- **Construção do grafo (cont.)**
  - 2 arcos por região x:



  - 2 arcos por cada ligação direta entre regiões x e y:

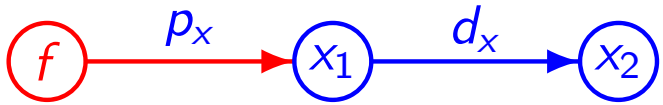
    (x_2, y_1) e (y_2, x_1), ambos com capacidade $+\infty$



  - Por cada arco referido, existe o arco inverso com capacidade 0

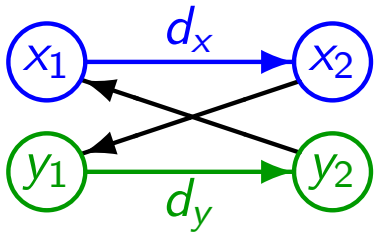
B - Rescue by Rail

- **Construção do grafo (cont.)**
  - 2 arcos por região x:



  - 2 arcos por cada ligação direta entre regiões x e y:

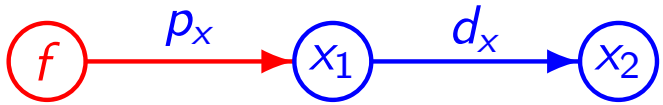
    (x_2, y_1) e (y_2, x_1), ambos com capacidade $+\infty$



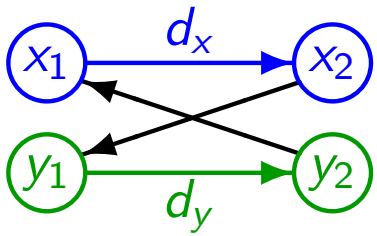
  - Por cada arco referido, existe o arco inverso com capacidade 0
- **Vértices especiais:** fonte f e dreno S_1 (chegada à região segura)

B - Rescue by Rail

- **Construção do grafo (cont.)**
  - 2 arcos por região x:



  - 2 arcos por cada ligação direta entre regiões x e y:
    (x_2, y_1) e (y_2, x_1), ambos com capacidade $+\infty$



  - Por cada arco referido, existe o arco inverso com capacidade 0
- **Vértices especiais:** fonte f e dreno S_1 (chegada à região segura)
- **Algoritmos:** Qualquer algoritmo clássico que calcule um fluxo máximo entre dois vértices seria aceite (e.g. Edmonds-Karp), porque $|V| \leq 2\,001$ e $|E| \leq 24\,000$

H - No More Stacking

Problema

- **Input:** Informação de número de voltas, pilotos e duração de pneus
- **Output:** Quantas estratégias *non-stacking*? (um único inteiro)

Limites

- $5 \leq N \leq 50$
- $0 < D \leq 20$
- $1 < T \leq 5$
- $1 < S_i \leq N$

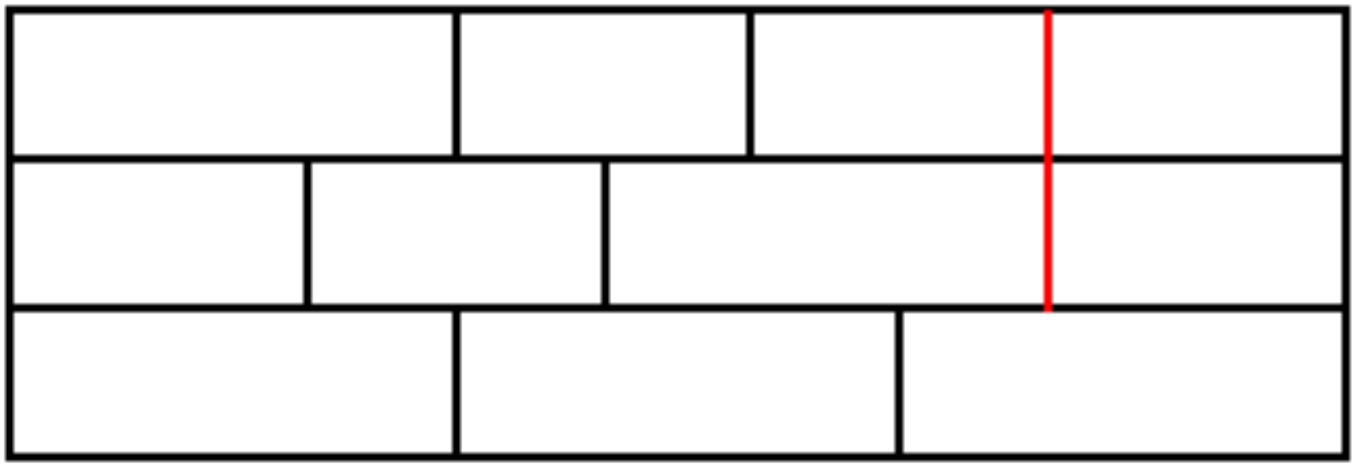
Classificação

- **Categorias:** programação dinâmica
- **Dificuldade:** Difícil

H - No More Stacking

Stacking

- Fórmula 1: os dois pilotos da equipa chegam ao mesmo tempo à box
- Neste problema:
  - equipas com mais que dois pilotos
  - stacking se dois pilotos **consecutivos** chegam ao mesmo tempo à box
- Numa imagem:

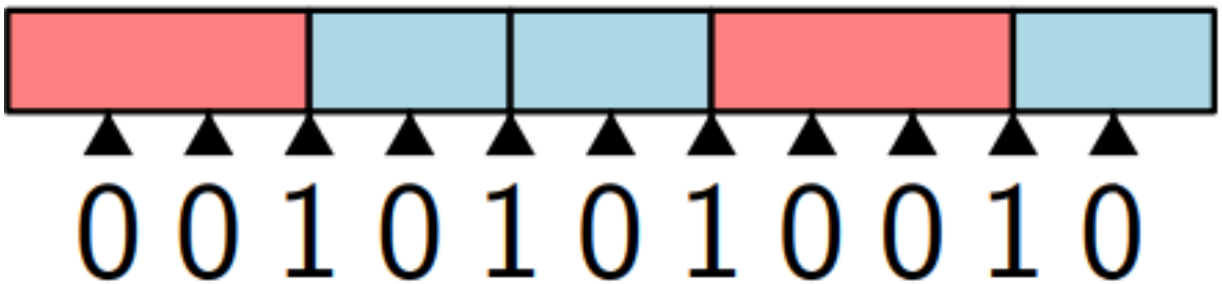


Q: Quantas estratégias *non-stacking* para uma equipa?

H - No More Stacking

Todas as estratégias de montagem de pneus

- As diferentes classes de pneus têm durações diferentes
- Cada pneu deve ser utilizado até ao máximo da sua capacidade
- É possível gerar **todas** as estratégias de montagem de pneus possíveis
  - dado o número de voltas da corrida
  - dada a duração de cada classe de pneus
- Como representar uma estratégia? Sugestão: **codificação binária**.
- Exemplo:
  - corrida de 12 voltas
  - pneus que duram 2 voltas (S); pneus que duram 3 voltas (M)
  - estratégia possível: M-S-S-M-S
  - codificação binária: $00101010010_2 = 338$



H - No More Stacking

Cálculo de estratégias non-stacking

- Calcular recursivamente $W(s, d)$
  - número de estratégias *non-stacking* para d pilotos
  - o último piloto utiliza a estratégia s
- Caso base:

$$W(s, 1) = 1$$

- Caso geral:

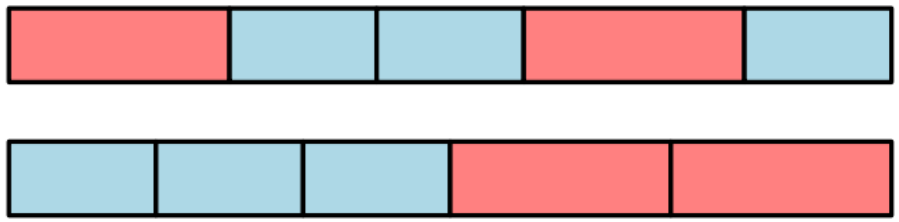
$$W(s, d) = \sum_{\text{s' comptavível com s}} W(s', d - 1)$$

- Como verifiar compatibilidade? **Operação AND lógico**.

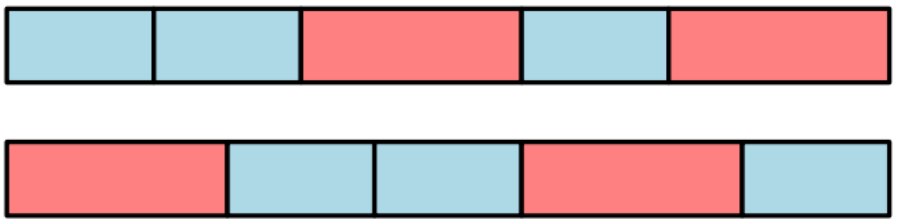
H - No More Stacking

Estratégias compatíveis

- Dadas duas estratégias s e s'
  - compatíves se $s \,\&\, s' = 0$



  - incompatíveis se $s \,\&\, s' \neq 0$



H - No More Stacking

Cálculo total de soluções

- Estratégias *non-stacking*, dados s e d

$$W(s,d) = \begin{cases} 1 & \text{se } d = 1 \\ \sum_{s \,\&\, s'=0} W(s', d-1) & \text{se } d > 1 \end{cases}$$

- Cálculo total de soluções
  - D pilotos
  - $\mathcal{S}$ conjunto de todas as estratégias de montagem de pneus

$$T(\mathcal{S}, D) = \sum_{s \in \mathcal{S}} W(s, D)$$

- Cálculo eficiente: **memorizar resultados intermédios**
  - compatibilidade do mesmo par de estratégias verificada repetidas vezes
  - calcular a função W usando programação dinâmica (matriz)