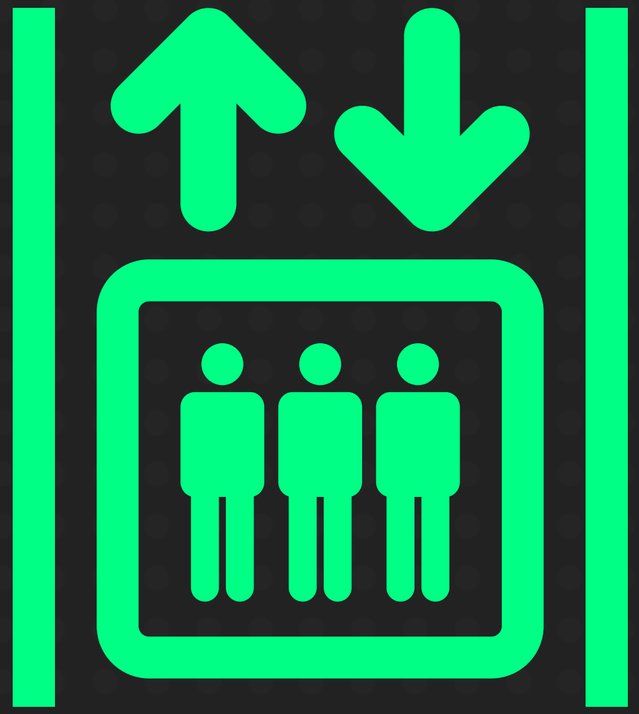
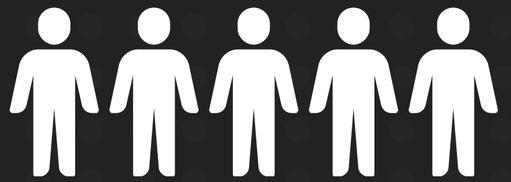
IPC-2011

ABRIL / 9

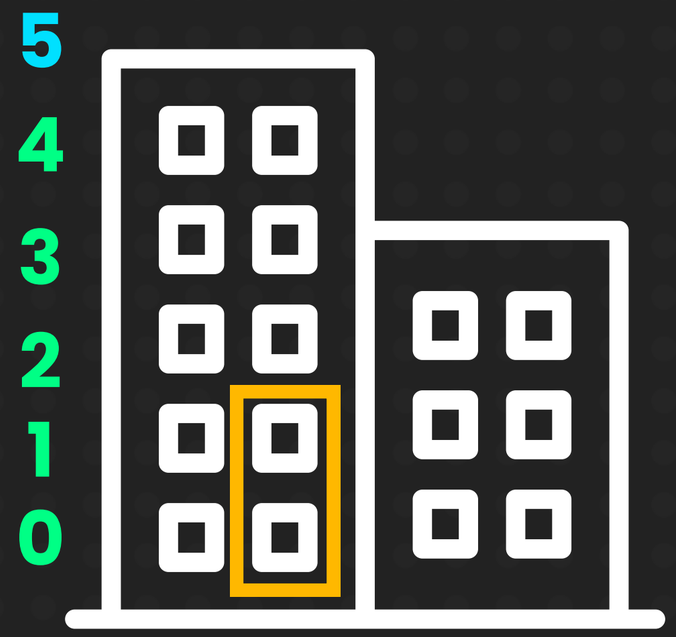
ELEVATOR SEQUENTIAL OPTIMAL





FUNDAMENTOS LÓGICOS DE INTELIGÊNCIA ARTIFICIAL

Sobre o problema



Há um prédio com N + 1 andares, numerados de 0 a N.

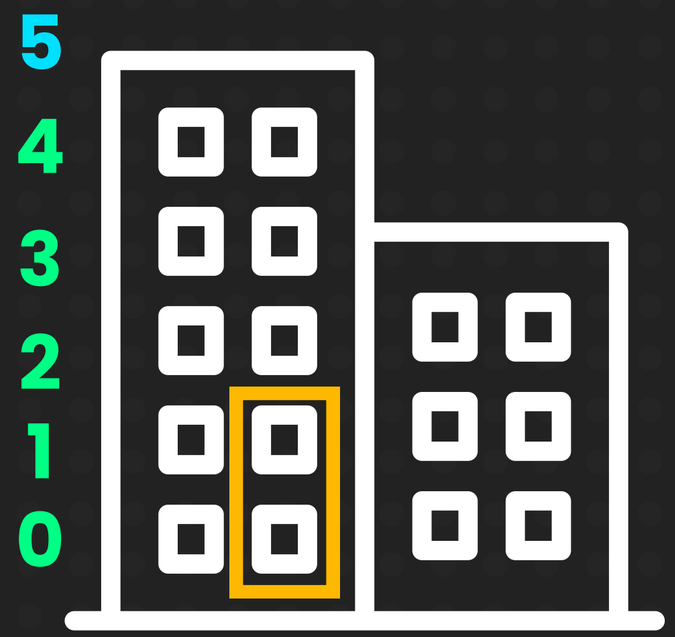
O prédio pode ser dividido em blocos de tamanho M, onde M divide N

Exemplo:

N = 4

M = 2

Sobre o problema



Blocos adjacentes têm um andar em comum.

Por exemplo, suponha que N = 12 e M = 4, então temos um total de **13** andares **(variando de 0 a 12)**, que formam 3 blocos de 5 andares cada, sendo **0 a 4, 4 a 8 e 8 a 12.**

Sobre o problema

O prédio possui K elevadores **rápidos** (acelerados) que **param apenas em andares que são múltiplos de M / 2**

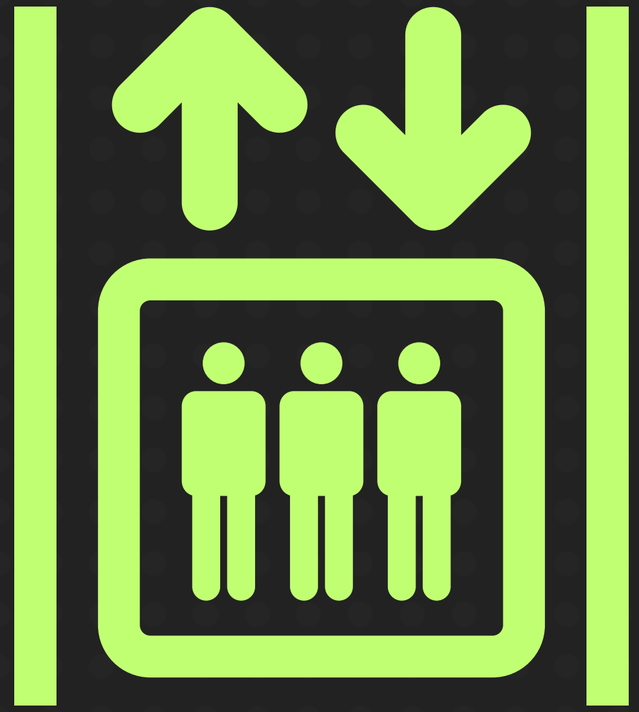
M deve ser um número par!

O elevador rápido tem capacidade para **X** pessoas

Sobre o problema

Dentro de cada bloco, há L elevadores lentos, que param em cada andar do bloco.

O elevador lento tem capacidade para **Y** pessoas



Requirements e Types

```
(:requirements :typing :action-costs)
(:types        elevator - object
                        slow-elevator fast-elevator - elevator
                        passenger - object
               count - object
       )
```

Predicados

```
(:predicates
        (passenger-at ?person - passenger ?floor - count)
        (boarded ?person - passenger ?lift - elevator)
        (lift-at ?lift - elevator ?floor - count)
        (reachable-floor ?lift - elevator ?floor - count)
        (above ?floor1 - count ?floor2 - count)
        (passengers ?lift - elevator ?n - count)
        (can-hold ?lift - elevator ?n - count)
        (next ?n1 - count ?n2 - count)
)
```

Predicados

(passenger-at ?person - passenger ?floor - count)

faz referencia/seta qual andar o passageiro está

(boarded ?person - passenger ?lift - elevator)

coloca uma pessoa **p** em um elevador **lift**

(lift-at ?lift - elevator ?floor - count)

mover os elevador entre os andares

Predicados

(reachable-floor ?lift - elevator ?floor - count)

marca em quais andares cada elevador pode ir

(above ?floor1 - count ?floor2 - count)

marca se **n1** está abaixo de **n2**

(passengers ?lift - elevator ?n - count)

quantos passageiros tem no elevador

Predicados

(can-hold ?lift - elevator ?n - count)

quantos passageiros um elevador pode levar

(next ?n1 - count ?n2 - count)

determina a ordem nos numeros, de **n1** até **nn**

Funções

```
(:functions (total-cost) - number
            (travel-slow ?f1 - count ?f2 - count) - number
            (travel-fast ?f1 - count ?f2 - count) - number
)
```

Funções

functions (total-cost) - number

Representa o custo total ao longo do planejamento

(travel-slow ?f1 - count ?f2 - count) - number

Custo de uma viagem entre andares de um elevador lento

recebe os parâmetros f1 e f2 ,que são os andares de inicio e de termino da viagem, respectivamente e retorna o custo da viagem

Funções

(travel-fast ?f1 - count ?f2 - count) - number

Custo de uma viagem entre andares de um elevador rápido

recebe os parâmetros f1 e f2 ,que são os andares de inicio e de termino da viagem, respectivamente e retorna o custo da viagem

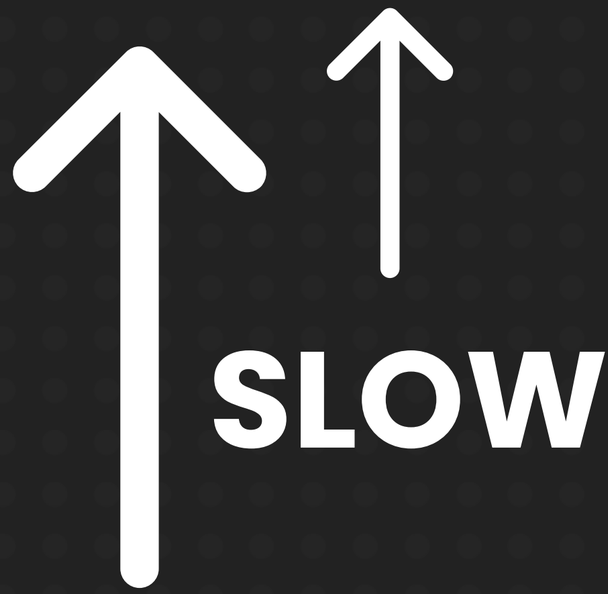
Funções

ARQUIVO DE PROBLEMA Custo das ações

```
(= (travel-fast n0 n3) 10) (= (travel-fast n0 n6) 19)

(= (travel-fast n3 n6) 10) (= (travel-fast n3 n9) 19)

(= (travel-fast n6 n9) 10) (= (travel-fast n6 n12) 19)

(= (travel-fast n9 n12) 10)

(= (total-cost) 0)
```

Actions

```
(:action move-up-slow
 :parameters (?lift - slow-elevator ?f1 - count ?f2 - count )
 :precondition (and (lift-at ?lift ?f1) (above ?f1 ?f2 ) (reachable-floor ?lift ?f2) )
 :effect (and
            (lift-at ?lift ?f2)
            (not (lift-at ?lift ?f1))
            (increase (total-cost)
            (travel-slow ?f1 ?f2))))
```



Actions

```
(:action move-down-slow
 :parameters (?lift - slow-elevator ?f1 - count ?f2 - count  )
 :precondition (and (lift-at ?lift ?f1) (above ?f2 ?f1 ) (reachable-floor ?lift ?f2) )
 :effect (and
            (lift-at ?lift ?f2)
            (not (lift-at ?lift ?f1))
            (increase (total-cost)
            (travel-slow ?f2 ?f1))))
```

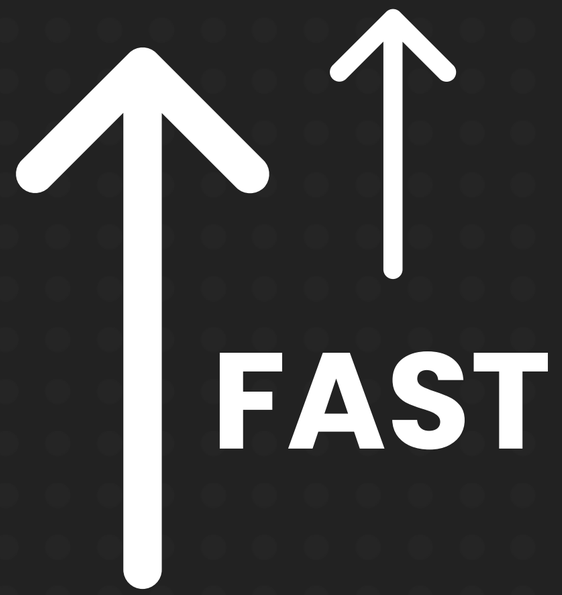
diferença do travel-slow e above

SLOW

Actions

```
(:action move-up-fast
 :parameters (?lift - slow-elevator ?f1 - count ?f2 - count )
 :precondition (and (lift-at ?lift ?f1) (above ?f1 ?f2 ) (reachable-floor ?lift ?f2) )
 :effect (and
            (lift-at ?lift ?f2)
            (not (lift-at ?lift ?f1))
            (increase (total-cost)
            (travel-slow ?f1 ?f2))))
```

igual ao slow



Actions

```
(:action move-down-fast
 :parameters (?lift - slow-elevator ?f1 - count ?f2 - count  )
 :precondition (and (lift-at ?lift ?f1) (above ?f2 ?f1 ) (reachable-floor ?lift ?f2) )
 :effect (and
                (lift-at ?lift ?f2)
                (not (lift-at ?lift ?f1))
                (increase (total-cost)
                (travel-slow ?f2 ?f1))))
```

igual ao slow também!

FAST

Actions

```
(:action board
 :parameters (?p - passenger ?lift - elevator ?f - count ?n1 - count ?n2 - count)
 :precondition (and
                    (lift-at ?lift ?f)
                    (passenger-at ?p ?f)
                    (passengers ?lift ?n1)
                    (next ?n1 ?n2)
                    (can-hold ?lift ?n2))
 :effect (and
                (not (passenger-at ?p ?f)
                (boarded ?p ?lift)
                (not (passengers ?lift ?n1))
                (passengers ?lift ?n2)))
```

N1 e N2 fazem referência a quantidade de pessoas no elevador.
N1 quantidade atual
N2 quantos vai ter

Actions

```
(:action leave
 :parameters (?p - passenger ?lift - elevator ?f - count ?n1 - count ?n2 - count)
 :precondition (and
                    (lift-at ?lift ?f)
                    (boarded ?p ?lift)
                    (passengers ?lift ?n1)
                    (next ?n2 ?n1))
 :effect (and
                (passenger-at ?p ?f)
                (not (boarded ?p ?lift))
                (not (passengers ?lift ?n1))
                (passengers ?lift ?n2)))
```

Resultado

Julia FF

```
(move-down-slow slow1-0 n8 n7)
(move-down-slow slow0-0 n6 n0)
(board p0 slow0-0 n0 n0 n1)
(board p1 slow0-0 n0 n1 n2)
(move-up-slow slow0-0 n0 n3)
(leave p0 slow0-0 n3 n2 n1)
(move-down-slow slow0-0 n3 n2)
(move-down-slow slow1-0 n7 n6)
(board p2 slow0-0 n2 n1 n2)
(move-up-slow slow0-0 n2 n6)
(leave p1 slow0-0 n6 n2 n1)
(leave p2 slow0-0 n6 n1 n0)
(board p2 slow1-0 n6 n0 n1)
(board p1 slow1-0 n6 n1 n2)
(move-up-slow slow1-0 n6 n7)
(leave p2 slow1-0 n7 n2 n1)
(move-up-slow slow1-0 n7 n11)
(leave p1 slow1-0 n11 n1 n0)
```

Referências

https://github.com/potassco/pddl-instances

https://planning.wiki/ref

Getting Started With Automated Planning - Dominik Schreiber

MUITO OBRIGADO!

Integrantes do grupo

Guilherme Peixoto
Luis Eduardo Lima
Rafael Carvalho
Renan Vieira