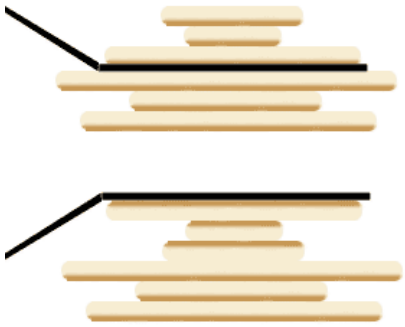
Ordenação das Pilha de Panquecas

Dada uma pilha de panquecas de vários tamanhos, podem ordená-las de modo descrescente, com a maior em baixo e a mais pequena no topo? Voçês possuem uma espátula com a qual podem inverter as i panquecas do topo, sendo i maior do que 1 e menor ou igual ao número de panquecas. Podemos inverter as duas de topo, ou as 3 de topo, ..., ou todas. A figura em baixo ilustra o problema com uma espátula com i=3; no topo espátula agarra 3 panquecas e no fundo aparecem invertidas:



Notem que desejamos inverter o menor número de panquecas, o que obriga a que o custo corresponda ao número de panquecas invertidas.

Formulação

Vamos representar os estados, as pilhas de panquecas, como um tuplo ordenado do topo para o fundo em que cada panqueca corresponde ao tamanho: a mais pequena é 1, a segunda mais pequena é a 2e por aí adiante; e assim o objectivo é o tuplo (1, 2, ..., *n*) e o estado inicial é (2, 1, 4, 6, 3, 5).

Implementação em Python

Escolhendo o tuplo para o estado, podemos escapar-nos de uma modelização mais purista Objecto Oriented e passar logo à classe do problema. Notem que não é necessário redefinir o ***equal()*** nem o ***hash()*** porque usamos um tuplo, nem o ***path_cost()*** porque o custo de cada acção é sempre 1, nem o goal_test() porque sabemos qual o estado final desejado.

In [1]:

```
from searchPlus import *

class PancakeProblem(Problem):

    def __init__(self, initial):
        """O objectivo é o tuplo ordenado por ordem crescente e o estado inicial é uma
 permutação de range(1, n+1).
        """
        self.initial, self.goal = tuple(initial), tuple(sorted(initial))

    def actions(self, state):
        """ As acções variam entre 2 e o número de panquecas
        """
        return range(2, len(state) + 1)

    def result(self, state, i):
        """Inverte-se as primeiras i panquecas e concatena-se com as que sobram
        """
        return state[:i][::-1] + state[i:]

    def path_cost(self, state1,action,next_state):
        """ A acção corresponde ao custo, porque inverter as n panquecas de topo
        corresponde a inverter n panquecas.
        """
        return action
```

Vamos criar um objecto do tipo **PancakeProblem**

In [2]:

```
c0 = PancakeProblem((2, 1, 4, 6, 3, 5))
```

Vamos verificar qual o estado inicial:

In [3]:

```
c0.initial
```

Out[3]:

```
(2, 1, 4, 6, 3, 5)
```

e final:

In [4]:

```
c0.goal
```

Out[4]:

```
(1, 2, 3, 4, 5, 6)
```

Quais as acções que se podem aplicar ao estado inicial?

In [5]:

```
acs=c0.actions(c0.initial)
print(acs)
```

```
range(2, 7)
```

Qual a primeira dessas acções?

In [6]:

```
acs[0]
```

Out[6]:

```
2
```

A última?

In [7]:

```
acs[-1]
```

Out[7]:

```
6
```

Vamos inverter as 3 panquecas do topo, partindo do estado inicial

In [10]:

```
print('no início:',c0.initial)
e1 = c0.result(c0.initial,3)
print('Invertemos as 3 de topo',e1)
print("com custo:", c0.path_cost(c0.initial,3,e1))
```

```
no início: (2, 1, 4, 6, 3, 5)
Invertemos as 3 de topo (4, 1, 2, 6, 3, 5)
com custo: 3
```