ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

IGOR PONTES TRESOLAVY

EP3 – ORDENAÇÃO DE PANQUECAS

SÃO PAULO
2021

SUMÁRIO

1. Como funciona seu algoritmo? Qual a ideia geral?

O algoritmo utilizado é do tipo *Branch and Bound*[1] e encontra sempre uma solução com o menor número possível de *flips* para qualquer pilha de panquecas. Para isso, utilizou-se métodos recursivos e *backtracking* na busca da solução.

Um dos algoritmos mais simples que resolve o problema simula o algoritmo de ordenação por seleção (*Selection Sort*[2]), mas utilizando somente operações de *flip* para fazê-lo. A ideia é dividir a pilha de panquecas em duas subpilhas: uma ordenada e outra desordenada. Inicialmente, a subpilha ordenada está vazia. Então, encontra-se a posição da panqueca de maior diâmetro na pilha desordenada; se não for a última posição de qualquer das duas subpilhas, flipa-se a pilha nessa posição e, finalmente, flipa-se a pilha na última posição da subpilha ordenada, inicialmente a posição 0 da pilha. Esse processo é repetido até a pilha estar completamente ordenada. Abaixo, há um delineado do algoritmo:

- Para cada elemento da pilha, menos o último:
    - Acha a posição da panqueca de maior diâmetro na subpilha desordenada
    - Se a posição encontrada não for a posição do elemento atual
        - Se a posição for a última posição da subpilha desordenada
            - Flipa as panquecas, na última posição da subpilha ordenada
        - Se a posição não for a última posição da pilha
            - Flipa as panquecas na posição
            - Flipa as panquecas, na última posição da subpilha ordenada

O algoritmo, apesar de relativamente rápido e simples, objetivamente nem sempre encontra a solução com o menor número de *flips*. Por exemplo, aplicando-o na pilha de 10 panquecas [6 7 5 2 3 9 5 8 10 4]:

flip(8): [6 7 5 2 3 9 5 8 4 10]

flip(0): [10 4 8 5 9 3 2 5 7 6]

flip(4): [10 4 8 5 6 7 5 2 3 9]

flip(1): [10 9 3 2 5 7 6 5 8 4]

flip(8): [10 9 3 2 5 7 6 5 4 8]

flip(2): [10 9 8 4 5 6 7 5 2 3]

flip(6): [10 9 8 4 5 6 3 2 5 7]

flip(3): [10 9 8 7 5 2 3 6 5 4]

flip(7): [10 9 8 7 5 2 3 4 5 6]

flip(4): [10 9 8 7 6 5 4 3 2 5]

flip(5): [10 9 8 7 6 5 2 3 4 5]

flip(6): [10 9 8 7 6 5 5 4 3 2]

O algoritmo ordena a pilha com 12 *flips*. No entanto, as menores soluções possuem somente 8 *flips*:

flip(5): [6 7 5 2 3 4 10 8 5 9] flip(0): [10 9 8 5 4 3 2 5 7 6]

flip(7): [6 7 5 2 3 4 10 9 5 8] flip(8): [10 9 8 5 4 3 2 5 6 7]

flip(8): [6 7 5 2 3 4 10 9 8 5] flip(3): [10 9 8 7 6 5 2 3 4 5]

flip(6): [6 7 5 2 3 4 5 8 9 10] flip(6): [10 9 8 7 6 5 5 4 3 2]

Portanto, um algoritmo melhor se mostra necessário para encontrar uma solução com o menor número de *flips* para qualquer pilha de panquecas.

Utilizar um algoritmo de *backtracking* recursivo é uma opção viável. No caso, para uma pilha de panquecas de tamanho arbitrário, o algoritmo realiza todas as combinações de *max* ou menos *flips* e a primeira menor solução encontrada é a solução devolvida, sendo *max* um número máximo de *flips* permitido estipulado incialmente e durante a execução do algoritmo, o que o caracteriza como um algoritmo de *Branch and Bound*, pois o limite de flips permitidos é atualizado conforme a execução do algoritmo.

Dado que achar a menor quantidade de *flips* para ordenar uma pilha de panquecas se trata de um problema NP-Difícil[3], heurísticas e parametrizações precisam ser utilizados no desenvolvimento de um algoritmo para que uma resolução razoavelmente eficiente do problema seja possível.

Abaixo, encontra-se um delineado do algoritmo:

- Se a quantidade de *flips* atual for menor que o máximo estipulado*
    - Se a pilha de panquecas não estiver resolvida
        - Para cada posição da pilha, menos a última
            - Se a posição atual não foi a última a ser flipada
                - Flipa a posição
                - Contabiliza e armazena o flip
                - Recursão
                - Desflipa a posição
                - Descontabiliza o f*lip*
    - Se a pilha de panquecas estiver resolvida
        - Se for a primeira solução encontrada
            - Guarda os *flips* e a quantidade dos mesmos
            - Atualiza o máximo estipulado
        - Se não for a primeira solução encontrada
            - Se a quantidade de *flips* for menor que a da última solução
                - Guarda os *flips* e a quantidade dos mesmos
                - Atualiza o máximo estipulado

*máximo estipulado inicial *(n* é o número de panquecas, *max* é o máximo estipulado incial):

n	max	n	max
1	1	11	13
2	2	12	14
3	3	13	15
4	4	14	16
5	5	15	17
6	7	16	18
7	8	17	19
8	9	18	20
9	10	19	22
10	11	>= 20	floor(18n/11)

Figura 1 - número máximo de *flips* permitido como função do número de panquecas na pilha

Os números máximos de *flips* necessários para qualquer pilha de $n < 20$ panqueca(s) já foram descobertos e são utilizados como parâmetros do algoritmo[4][5]. Para $n >= 20$, o resultado da operação $18n/11$ arredondado para baixo (pela função *floor()*[6]) é utilizado com parâmetro inicial. Essa expressão é utilizada pois sabe-se que, para qualquer número de panquecas em uma pilha, a menor solução sempre terá menos *flips* que o resultado dessa operação[7].

A heurística utilizada dita que, ao encontrar uma solução, dado que o objetivo é sempre encontrar uma menor solução e não todas as soluções, esse máximo estipulado é atualizado de modo que o algoritmo só buscará soluções de tamanho menor que a última solução encontrada. Isso permite que o algoritmo consiga encontrar soluções ótimas mais rapidamente, dado que, como demonstrado a seguir, o algoritmo tende a ser pouco eficiente para pilhas de panquecas grandes.

2. Qual sua estimativa de complexidade de tempo de seu algoritmo para ordenar *n* panquecas?

Para *n's* arbitrariamente grandes, o algoritmo começa com $(n - 1)(n - 2)^{floor(18n/11) - 1}$ *flips* possíveis ("– 1" por conta da última posição da pilha e "– 2" por conta da última posição da pilha e da posição da última jogada feita). Portanto, a complexidade é de $O((n - 2)^{floor(18n/11)})$, pois, para *n's* grandes, $(n - 1) \sim (n - 2)$ e $(n - 1)(n - 2)^{floor(18n/11) - 1} \sim (n - 2)^{floor(18n/11)}$.

A complexidade diminui quando uma solução que requer menos de $floor(18n/11)$ *flips* é encontrada. No entanto, como visto pela notação *O* grande, a complexidade inicial do algoritmo aumenta com n elevado a ele mesmo, multiplicado por uma constante, o que indica que a complexidade de tempo aumenta significativamente conforme *n* aumenta, especialmente se soluções pequenas não forem encontradas cedo na busca.

3. Qual o número mínimo e máximo de *flips* seu algoritmo faz para ordenar uma sequência?

O número mínimo de *flips* é 0, o que ocorre quando a pilha de panquecas de entrada já está ordenada e não há nenhum movimento necessário para ordenar a pilha.

O número máximo de *flips* ocorre quando a pilha de panquecas desordenada incial só pode ser resolvida com *floor*(18*n*/11) *flips*, para *n* >= 20, ou com a quantidade máxima inicial de *flips* estipulada, para *n* < 20, cujos valores encontram-se na Figura 1.

Nesse caso, o algoritmo irá testar algumas combinações de *floor*(18*n*/11) *flips* e todas as combinações de *floor*(18n/11) - 1 ou menos *flips*, até concluir que a pilha pode ser resolvida somente com *floor*(18n/11) *flips* e devolver a primeira solução encontrada.

4. Mostre sequências em que seu algoritmo realiza o número máximo e mínimo de *flips*.

Para a pilha [3 2 1], *max* = 3, mas não há movimentos necessários.

Para a pilha [2 3 1], *max* = *3*:

flip(0): [1 3 2]
flip(1): [1 2 3]
flip(0): [3 2 1]

O algoritmo realiza o número máximo de *flips* estipulado inicialmente.

Uma variante interessante do problema é quando as panquecas têm um lado mais queimado que deve ficar para baixo. Ou seja, não basta ordenar as panquecas, mas manter a parte que estava inicialmente para cima desta forma. Seu algoritmo resolve o problema com esta restrição extra?

O algoritmo, da maneira que está implementado, não resolve o problema das panquecas queimadas. No entanto, com duas modificações no algoritmo anterior, é possível obter um algoritmo de ordenação de panquecas queimadas. As mudanças em relação ao algoritmo anterior estão em negrito no delineado abaixo:

- Se a quantidade de *flips* atual for menor que o **máximo estipulado**
    - Se a pilha de panquecas não estiver resolvida
        - Para cada posição da pilha, **inclusive** a última
            - Se a posição atual não foi a última a ser flipada
                - Flipa a posição
                - Contabiliza e armazena o flip
                - Recursão
                - Desflipa a posição
                - Descontabiliza o f*lip*
    - Se a pilha de panquecas estiver resolvida
        - Se for a primeira solução encontrada
            - Guarda os *flips* e a quantidade dos mesmos
            - Atualiza o máximo estipulado
        - Se não for a primeira solução encontrada
            - Se a quantidade de *flips* for menor que a da última solução
                - Guarda os *flips* e a quantidade dos mesmos
                - Atualiza o máximo estipulado

Para esse algoritmo, o máximo estipulado inicial é sempre 2*n*, pois sabe-se que qualquer pilha de *n* panquecas queimadas pode ser ordenada com 2*n* ou menos movimentos[8]. Além disso, a última panqueca está disponível para ser flipada, pois, para esse problema, o lado da panqueca que está virado para cima não é desimportante e, por isso, flipar a última panqueca individualmente se faz necessário.

A complexidade desse algoritmo, obtida de maneira análoga à anterior, é maior que a do algoritmo do problema anterior, como esperado: $O((n - 1)^{2n})$. Porém, esse algoritmo eventualmente também sempre acha a solução ótima para qualquer pilha de panquecas queimadas.

Referências:

[1] Sobre algoritmos *Branch and Bound*: Wikipedia – Branch and Bound

[2] Sobre algoritmos *Selection Sort:* Wikipedia – Selection Sort

[3] Ordenação de panquecas é NP-Difícil: Pancake Flipping is Hard

[4] Número máximo de *flips* necessários para n <= 17: Improved Pancake Sorting

[5] Número máximo de *flips* necessários para 18 <= n <= 19: The OEIS Foundation - A058986

[6] Função *floor* ou "piso": Wikipedia – Parte Inteira

[7] Limite de 18*n*/11 *flips*: An (18/11)*n* upper bound for sorting by prefix reversals

[8] 2*n flips* é o máximo necessário para qualquer pilha de panquecas queimadas: On the problem of sorting burnt pancakes

Outras referências:

- Vídeo sobre ordenação de panquecas: Pancake Numbers – Numberphile
- Algoritmo simples para a ordenação de panquecas: Pancake Sorting – Geeksforgeeks
- Versão do Algoritmo simples de ordenação de panquecas para o problema das panquecas queimadas: Burnt Pancakes – Dodona