Análise de Algoritmos - T3

Renato Rodrigues Vandré Leal

Universidade Federal de Uberlândia

13/12/2018

Introdução

Recapitulando, o objetivo principal é inserir itens contendo pesos/tamanhos em caixas de determinada capacidade minimizando ao máximo a quantidade de caixas utilizadas.

Exemplos de Uso

- Alocação de memória
- Alocação de artigos de jornal nas páginas
- Carregamento de cargas de caminhões
- Alocação dos comerciais de tv dentro dos intervalos corretos de programas

Tipos de Algoritmos

- **Online**: sem nenhum conhecimento prévio.
- **Offline**: com conhecimento prévio.
- **Semi-online**: conhecimento de apenas algumas informações.

Definições

- Item grande $> 1/2$
- Item pequeno $\leq 1/2$
- Caixa aberta
- Caixa fechada

Online: First-Fit e Best-Fit

Os algoritmos First-Fit (FF) e Best-Fit (BF) nunca fecham uma caixa aberta antes que todos os itens são encaixotados, tornando a complexidade de espaço O(n). Ambos algoritmos podem ser implementados em tempo O(n log n).

Offline: First-Fit Decreasing e Best-Fit Decreasing

Os algoritmos First-Fit Decreasing (FFD) e Best-Fit Decreasing (BFD) são dois dos melhores algoritmos offline. Ambos ordenam os items de forma decrescente antes que os algoritmos FF e BF sejam aplicados.

As propostas dos autores

Um algoritmo offline com pior caso absoluto em uma razão de 3/2 operante em tempo linear e complexidade de espaço constante.

Outro algoritmo online também com tempo linear e complexidade de espaço constante. Porém com pior caso absoluto em uma razão de 7/4.

Algoritmo 1 - Caso Offline

1. Colocar cada item grande em uma caixa. Indexar as caixas não-cheias em ordem arbitrária. Definir todas essas caixas como caixas **ativas**. Repetidamente organizar os itens pequenos da seguinte forma.
2. Se houver uma caixa **ativa e aberta**, colocar o item corrente **a1** na caixa com o menor índice se a caixa tiver espaço suficiente para **a1**. Caso contrário, fechar esta caixa **ativa** e considerar uma caixa **extra** como segue:
    - Se houver uma caixa **extra aberta** e ela tiver espaço para **a1**, colocar a1 nela.
    - Caso contrário, definir a **caixa extra** como fechada. Abrir uma nova caixa para **a1** e definir essa caixa como uma **caixa extra**.
3. Se não houver nenhuma caixa **ativa e aberta**, criar uma nova caixa para **a1** e definir esta nova caixa como uma **caixa ativa**.

Algoritmo 1 - Exemplo

- $s(a1) = 0.8$;
- $s(a2) = 0.6$;
- $s(a3) = 0.3$;
- $s(a4) = 0.4$;
- $s(a5) = 0.2$;
- $s(a6) = 0.3$;
- $s(a7) = 0.4$;
- $s(a8) = 0.5$;
- $s(a9) = 0.2$.

Algoritmo 1 - Exemplo

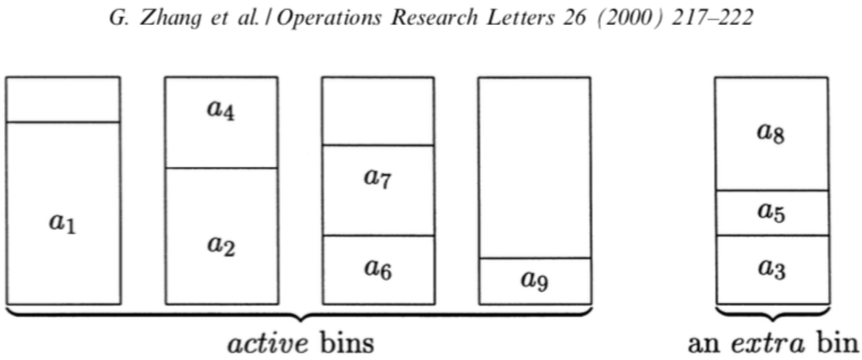

Fig. 1.

Algoritmo 1 - Observação

O algoritmo 1 é um algoritmo **semi-online** visto que apenas necessita da informação dos itens grandes antes da execução. Caso não haja itens grandes, o algoritmo se torna um **algoritmo online**.

Algoritmo 1 - Teorema

Teorema: possui tempo linear com pior caso absoluto 3/2.

Prova: O algoritmo possui tempo linear uma vez que cada item é empacotado em tempo constante. O processo de empacotamento consiste em dois conjuntos de caixas X e Y, onde X contém as caixas ativas e Y o conjunto de caixas extras. Todos os itens em caixas extras são pequenos. Com exceção da última caixa extra, todas as caixas extras contém pelo menos dois items. E somente existem uma caixa ativa e uma extra por vez, por isso o espaço constante.

Algoritmo 1 - Prova

X: Conjunto de caixas ativas;
Y: Conjunto de caixas extras;
N_a**:** Número total de caixas ativas;
N_c**:** Número total de caixas fechadas;
N_e**:** Número total de caixas extras;

Uma vez que todas a caixas extras contém apenas itens pequenos, todas elas terão no mínimo dois itens, com excessão da última. Implicando em $N_e \leq \lceil \mathrm{N}_c/2 \rceil$.

Algoritmo 1 - Prova

Case 1: $N_c \leqslant N_a - 1$. In this case,

$$A_1(L) = N_a + N_e \leqslant N_a + \lceil N_c/2 \rceil \leqslant N_a + \lceil (N_a - 1)/2 \rceil$$

$$\leqslant 3N_a/2.$$

If $N_c = N_a - 1$,

$$\mathrm{OPT}(L) \geqslant \sum_{i=1}^{n} s(a_i) = \sum_{a_i \in X} s(a_i) + \sum_{a_i \in Y} s(a_i)$$

$$> N_c = N_a - 1,$$

Algoritmo 1 - Prova

Case 2: $N_c = N_a$. In this case,

$$A_1(L) = N_a + N_e \leqslant N_a + \lceil N_c/2 \rceil = N_a + \lceil N_a/2 \rceil$$

$$\leqslant 3N_a/2 + \tfrac{1}{2}.$$

On the other hand,

$$\mathrm{OPT}(L) \geqslant \sum_{i=1}^{n} s(a_i) = \sum_{a_i \in X} s(a_i) + \sum_{a_i \in Y} s(a_i) > N_a,$$

i.e., $\mathrm{OPT}(L) \geqslant N_a + 1$. $A_1(L)/\mathrm{OPT}(L) \leqslant \frac{3}{2}$ also holds.
We have proven $R_{A_1} \leqslant \frac{3}{2}$.

Algoritmo 2 - Caso Online

1. Abrir uma caixa para o primeiro item e definí-la como **ativa**.
2. Se existir uma caixa ativa **B1 aberta**, colocar o item corrente **a1** dentro de **B1** se esta tiver espaço suficiente para o item. Supondo que esta caixa não tenha espaço para **a1**:
   - Se **a1** for um item grande, criar uma nova caixa para ele. Se existirem duas caixas-L, fechá-las.
   - Se **a1** for um item pequeno, fechar a caixa **ativa**. Definir uma **caixa-L aberta** como **ativa**, caso exista. Se existir uma caixa **extra aberta** e tal caixa tiver espaço suficiente para a1, colocar o item na caixa. Do contrário, abrir uma nova caixa para a1 e defini-la como uma **caixa extra**.
3. Se não existir nenhuma **caixa ativa e aberta**, criar uma nova caixa para o item **a1** e defini-la como **caixa ativa**.

Algoritmo 2 - Exemplo

- s(a3) = 0.3;
- s(a4) = 0.4;
- s(a5) = 0.2;
- s(a2) = 0.6;
- s(a1) = 0.8;
- s(a6) = 0.3;
- s(a7) = 0.4;
- s(a8) = 0.5;
- s(a9) = 0.2.

Algoritmo 2 - Exemplo

G. Zhang et al. / Operations Research Letters 26 (2000) 217–222

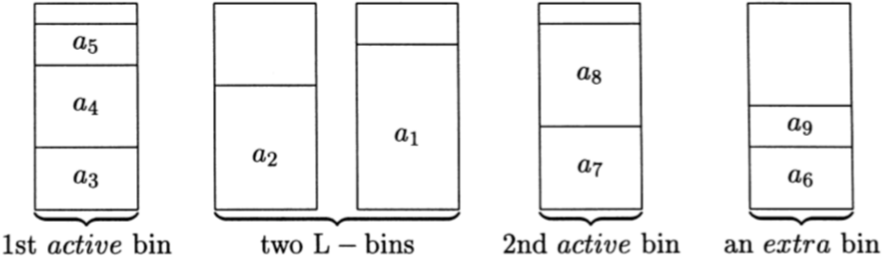

Fig. 2.

Algoritmo 2 - Teorema

Teorema: possui tempo linear com pior caso absoluto 7/4.

Prova: O algoritmo possui tempo linear uma vez que cada item é empacotado em tempo constante. O processo de empacotamento consiste em dois conjuntos de caixas X e Y, onde X contém as caixas ativas e Y o conjunto de caixas extras. Todos os itens em caixas extras são pequenos. Com exceção da última caixa extra, todas as caixas extras contém pelo menos dois items.

Fonte

Zhang, G., Cai, X. and Wong, C.K., 2000. Linear time-approximation algorithms for bin packing. Operations Research Letters, 26(5), pp.217-222. Zehmakan, A.N.,

2016. Bin Packing Problem: A Linear Constant-Space 3/2-Approximation Algorithm. arXiv preprint arXiv:1610.08820.