# Notas sobre Análise Amortizada

José Bernardo Barros

Algoritmos e Complexidade

## Motivação

Considere-se a seguinte função `inc` que, recebe como argumento um vector com os *bits* de um número, e altera esse vector de forma a representar o sucessor do número original.
Assim, por exemplo, para o vector representar o número 119 (cuja representação binária é `1110111`) deve ter todos os elementos a zero excepto os 7 últimos, que devem guardar a sequência `1 1 1 0 1 1 1`. A invocação da função `inc` com esse vector, deverá alterar apenas as 4 últimas posições do vector de forma a representar o número 120 (cuja representação binária é `1111000`).

```
int inc (int b[], int N) {
int i, r=0;

    i = N-1;
    while ((i >= 0) && (b[i] == 1)) {
        b[i] = 0;
        i--;
    }
    if (i>=0) b[i] = 1;
    else r = 0;

    return r;
}
```

A complexidade desta função pode ser reduzida ao número de bits alterados, que varia entre

- 1 no (**melhor**) caso de o bit menos significativo ser `0`
- $N$ no (**pior**) caso de os últimos $N-1$ bits serem `1`.

Uma análise do caso médio revela-nos que o número médio (ou esperado) de bits que são alterados é, no entanto, muito mais próxima do melhor caso apresentado.
Antes de proceder a esse cálculo, vejamos um exemplo para compreendermos o que queremos dizer com o custo médio (ou esperado) de uma operação. Façamos então este cálculo para um array com 4 posições ($N = 4$). A tabela seguinte mostra os vários valores que o array pode tomar (`b`), o resultado da operação (`b'`) e o custo (número de bits alterados) para cada um desses inputs possíveis.

| b | b' | **Custo** |
|---|---|---|
| 0000 | 0001 | 1 |
| 0001 | 0010 | 2 |
| 0010 | 0011 | 1 |
| 0011 | 0100 | 3 |
| 0100 | 0101 | 1 |
| 0101 | 0110 | 2 |
| 0110 | 0111 | 1 |
| 0111 | 1000 | 4 |

| b | b' | **Custo** |
|---|---|---|
| 1000 | 1001 | 1 |
| 1001 | 1010 | 2 |
| 1010 | 1011 | 1 |
| 1011 | 1100 | 3 |
| 1100 | 1101 | 1 |
| 1101 | 1110 | 2 |
| 1110 | 1111 | 1 |
| 1111 | 0000 | 4 |

Assumindo que os vários valores do input são igualmente prováveis, o custo médio é a média dos vários custos. No exemplo apresentado este custo é de 1.875 (30/16).
Note-se que o cálculo desta média pode ser feito

- somando todos os elementos e dividindo pelo número deles

$$\frac{1+2+1+3+1+2+1+4+1+2+1+3+1+2+1+4}{16}$$

- organizando as várias parcelas do númerador

$$\frac{8*1+4*2+2*3+2*4}{16} = \frac{1}{2}*1+\frac{1}{4}*2+\frac{1}{8}*3+\frac{1}{8}*4$$

Vejamos agora como fazer este cálculo para um vector de tamanho $N$.

- A assumpção de que todas os valores do input são igualmente prováveis, é equivalente a dizer que cada posição do vector pode ter, com a mesma probabilidade, os valores `0` ou `1`.

- Desta forma,
  - a probabilidade de mudar apenas 1 bit é $\frac{1}{2}$ (corresponde à probabilidade de o bit menos significativo ser `0`).
  - a probabilidade de mudar apenas 2 bits é $\frac{1}{4}$ (corresponde à probabilidade de o bit menos significativo ser `1` e o seguinte ser `0`).
  - a probabilidade de mudar apenas 3 bits é $\frac{1}{8}$ (corresponde à probabilidade de os 2 bits menos significativos serem `1` e o seguinte ser `0`).
  - a probabilidade de mudar apenas k bits ($0 \leq k < N$) é $\frac{1}{2^k}$ (corresponde à probabilidade de os $k-1$ bits menos significativos serem `1` e o seguinte ser `0`).
  - Existem dois casos em que o número de bits mudados é $N$: se todos os $N-1$ bits menos significativos forem `1`, fazem-se sempre $N$ alterações.

- O custo médio é por isso dado pela expressão

$$\overline{T}(N) = (\sum_{k=1}^{N} \frac{k}{2^k}) + \frac{N}{2^N} = 2 - \frac{1}{2^{N-1}}$$

É de notar que $\lim \overline{T}(N) = 2$ e que por isso

$$\overline{T}(N) = \Theta(1)$$

A razão de ser desta proximidade do caso médio e do melhor caso, ou por oposição, da diferença tão acentuada entre o caso médio e o pior caso, deve-se à pouca probabilidade de o pior caso acontecer.

Esta observação motiva o uso de uma outra forma de analisar a complexidade de algoritmos conhecida genericamente por **análise amortizada** e que consiste em, ao invés de analisar o custo de uma operação, analisar o custo de uma sequência de operações.
É costume utilizar como sequência de operações a analisar, a sequência de pior custo.

Dada então uma sequência de $N$ operações $\{op_i\}_{1 \leq i \leq N}$ cada uma com um custo $c_i$, pretendemos determinar o custo amortizado da operação $i$, denotado por $\widehat{c}_i$ de tal forma que a soma dos custos amortizados seja um limite superior da soma dos custos reais, i.e.,

$$\sum_{i=1}^{N} \widehat{c}_i \geq \sum_{i=1}^{N} c_i$$

# 1 Análise Agregada

Nesta abordagem à análise amortizada, aquilo que se faz é começar por calcular a soma dos custos reais das operações da sequência a analisar. Com este valor calculado, podemos estimar o custo amortizado de cada operação da sequência como

$$\widehat{c}_i = \frac{1}{N} \sum_{i=1}^{N} c_i$$

Com esta definição, a propriedade acima é trivialmente verificada:

$$\begin{array}{ll} \sum_{i=1}^{N} \widehat{c}_i & = \sum_{i=1}^{N} (\frac{1}{N} \sum_{j=1}^{N} c_j) \\ & = (\frac{1}{N} \sum_{j=1}^{N} c_j) \sum_{i=1}^{N} 1 \\ & = (\frac{1}{N} \sum_{j=1}^{N} c_j)\, N \\ & = \sum_{j=1}^{N} c_j \end{array}$$

Vejamos então como proceder para o caso que estamos a analisar (a função `inc` apresentada acima). Consideremos então uma sequência de $N$ invocações desta função a um mesmo vector `b` com um número suficientemente grande de posições e que está inicialmente todo preenchido a `0` (veremos que esta assumpção dos valores iniciais não é realmente importante na conclusão final).
A tabela sequinte (que já foi apresentada acima), mostra o custo das várias operações desta sequência.

| i | input | | | | | | | output | | | | | | | | $c_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 = 1 |
| 2 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 = 1+1 |
| 3 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 = 1 |
| 4 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | $\cdots$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 3 = 1+1+1 |
| 5 | $\cdots$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | $\cdots$ | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 = 1 |
| 6 | $\cdots$ | 0 | 0 | 0 | 0 | 1 | 0 | 1 | $\cdots$ | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 2 = 1+1 |
| 7 | $\cdots$ | 0 | 0 | 0 | 0 | 1 | 1 | 0 | $\cdots$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 = 1 |
| 8 | $\cdots$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | $\cdots$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 4 = 1+1+1+1 |
| ... | | | | | | | | | | | | | | | | | |

O que pretendemos calcular é a soma dos elementos da última coluna $\sum c_i$ para depois calcular a sua média.
Neste caso, e como já indicado na coluna, podemos ver que:

- todas as linhas têm a parcela 1
- metade das linhas tem também a parcela +1
- um quarto das linhas têm também mais uma parcela +1
- ...

Generalizando, a soma dos valores da última coluna pode ser calculada por:

$$N + \frac{N}{2} + \frac{N}{4} + ... = \sum_{i=0}^{?} \frac{N}{2^i}$$

O limite superior deste somatório corresponde ao número de vezes que consequimos dividir $n$ por 2, i.e., $\log_2 N$ (em rigor, este limite superior corresponde ao maior inteiro que seja menor ou igual a $\log_2 N$, i.e., $\lfloor \log_2 N \rfloor$).
Temos então o custo $C$ de fazer $N$ incrementos (sem overflow)

$$C_N = \sum_{i=0}^{\log_2 N} \frac{N}{2^i} = N \sum_{i=0}^{\log_2 N} \frac{1}{2^i} = N\,(2 - \frac{1}{2^{\log_2 N}}) = N\,(2 - \frac{1}{N}) = 2\,N - 1$$

Para sabermos agora o custo (amortizado) de cada operação, apenas temos que dividir este custo pelo número de operações em causa:

$$\widehat{c_i} = \frac{C_N}{N} = \frac{1}{N}(2\,N - 1) = 2 - \frac{1}{N} = \Theta(1)$$

## 2 Método Contabilístico

Muitas vezes não é fácil fazer o cálculo agregado dos custos de uma sequência de operações. Existem por isso outros métodos que ajudam, se não a calcular essa soma, pelo menos a determinar um limite superior razoável.

No método contabilistico da análise amortizada aquilo que vamos tentar é estimar um custo $\widehat{c_i}$ para a operação $i$ de tal forma que a soma dos custos das operações da sequência seja menor do que a soma destes custos esperados, i.e., devemos estimar o custo $\widehat{c_i}$ de tal forma que

$$\sum_{i=1}^{N} c_i \leq \sum_{i=1}^{N} \widehat{c_i}$$

Esta desigualdade pode ser escrita sob a forma

$$(\sum_{i=0}^{N} \widehat{c_i}) - (\sum_{i=0}^{N} c_i) \geq 0$$

Uma forma (mas não a única) de garantir que esta inequação é válida, é garantirmos que, para todo o $K$,

$$B_K \doteq (\sum_{i=1}^{K} \widehat{c_i}) - (\sum_{i=1}^{K} c_i) \geq 0$$

A analogia que vamos usar é a de uma conta bancária em que se pretende garantir um saldo positivo:

- o termo $D_K \doteq \sum_{i=1}^{K} \widehat{c_i}$ corresponde à soma de todos os *depósitos* (até à operação $K$).
- o termo $W_K \doteq \sum_{i=1}^{K} c_i$ corresponde à soma de todos os *levantamentos* (até à operação $K$).

Visto desta perspectiva, podemos definir $B_k$ como o *saldo* após a operação $k$ como

$$\begin{aligned} B_k &= D_k - W_k \\ &= (\textstyle\sum_{i=0}^{k} \widehat{c_i}) - (\sum_{i=0}^{k} c_i) \\ &= (\widehat{c_k} + \textstyle\sum_{i=0}^{k-1} \widehat{c_i}) - (c_k + \sum_{i=0}^{k-1} c) \\ &= (\textstyle\sum_{i=0}^{k-1} \widehat{c_i} - \sum_{i=0}^{k-1} c_i) + (\widehat{c_k} - c_k) \\ &= B_{k-1} + (\widehat{c_k} - c_k) \end{aligned}$$

O que nos dá uma forma de calcular o *saldo* após uma operação a partir do *saldo* antes e dos custos reais e estimados dessa operação.
Por razões de completude (e uma vez que a fórmula que obtivemos é uma recorrência) devemos indicar o valor inicial $B_0$. É costume usar 0 para este valor, embora não seja necessário para uma análise assimptótica (este valor inicial corresponde à assumpção feita na análise agregada de começar a sequência a analisar com um determinado valor).

Vejamos então como este método pode ser aplicado ao problema que já apresentámos (incremento).
A tabela apresentada atrás contem já a coluna correspondente aos custos reais (levantamentos). Na tabela seguinte vamos acrescentar duas novas colunas que correspondem aos custos estimados e ao saldo.
Para isso vamos estimar que o custo de cada operação é constante, i.e., $\widehat{c_i} = 2$ (nesta altura da exposição do problema já não há muitas surpresas sobre qual o custo amortizado de cada operação!).

Esta estimativa deve ser feita de forma a prever futuros custos adicionais das próximas operações. Neste caso, cada operação de `inc` transforma uma sequência de `1`s em `0`s e finalmente transforma um `0` em `1`. Se admitirmos que o custo de passar os `1`s a `0` se faz à custa do valor *amealhado*, devemos amealhar 1, para mais tarde o podermos usar para converter esse `1` (que acabou de ser produzido).

| **i** | input | | | | | | | | $\mathbf{c_i}$ | $\widehat{\mathbf{c_i}}$ | $\mathbf{B_i}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | 0 |
| 1 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 1 |
| 2 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 1 |
| 3 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 2 | 2 |
| 4 | $\cdots$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 3 | 2 | 1 |
| 5 | $\cdots$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 2 | 2 |
| 6 | $\cdots$ | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 2 | 2 | 2 |
| 7 | $\cdots$ | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 2 | 3 |
| 8 | $\cdots$ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 4 | 2 | 1 |
| 9 | $\cdots$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 2 | 2 |
| 10 | $\cdots$ | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 2 | 2 | 2 |
| 11 | $\cdots$ | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 2 | 3 |
| 12 | $\cdots$ | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 3 | 2 | 2 |
| 13 | $\cdots$ | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 2 | 3 |
| 14 | $\cdots$ | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 2 | 2 | 3 |
| 15 | $\cdots$ | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 2 | 4 |
| 16 | $\cdots$ | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 5 | 2 | 1 |
| 17 | $\cdots$ | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 2 |

Esta tabela mostra-nos que o valor estimado ($\widehat{c}_i = 2$) para cada operação é suficiente para garantir que o saldo nunca toma valores negativos. De facto, e tal como vimos atrás, em metade dos casos o custo real é 1, permitindo-nos *poupar* 1 por cada uma dessas operações (continuando na analogia contabilística, um *depósito* de 2 e um *levantamento* de 1 permite-nos *poupar* 1).

# 3 Método do Potencial

O método de análise amortizada visto na secção anterior baseia-se na estimativa de um custo de cada operação de tal forma que o somatório dos custos estimados seja superior à soma dos custos reais dessas operações.
No método do potencial vamos tentar calcular estes custos estimados a partir de uma função (dita de potencial) sobre a estrutura de dados (estado) em questão.

Suponhamos então que definimos uma função $\Phi$ que, mapeia cada estado (no caso que temos vindo a analisar, um vector de `0`s e `1`s) num número (real) e que satisfaz as seguintes propriedades:

- $\Phi(S) \geq 0$ para todos os possíveis estados $S$
- $\Phi(S_0) = 0$ em que $S_0$ corresponde ao estado inicial.

Usando esta função de potencial, vamos definir o custo estimado de cada operação da sequência $(\widehat{c_i})$ como

$$\widehat{c_i} = c_i + (\Phi(S_i) - \Phi(S_{i-1}))$$

em que $c_i$ corresponde ao custo real da operação $i$ e $S_{i-1}$ e $S_i$ correspondem aos estados antes e depois de executar a operação $i$.

Por simplicidade de notação, vamos passar a denotar $\Phi(S_i)$ apenas por $\Phi_i$

Vejamos agora como estas definições nos permitem obter um majorante para o custo de $N$ operações.

$$\begin{aligned}
\sum \widehat{c_i} &= \widehat{c_1} + \widehat{c_2} + \cdots + \widehat{c_{N-1}} + \widehat{c_N} \\
&= (c_1 + \Phi_1 - \Phi_0) + (c_2 + \Phi_2 - \Phi_1) + \cdots + (c_N + \Phi_N - \Phi_{N-1}) \\
&= c_1 + \Phi_1 - \Phi_0 + c_2 + \Phi_2 - \Phi_1 + \cdots + c_N + \Phi_N - \Phi_{N-1} \\
&= c_1 + c_2 + \cdots + c_N + \Phi_N - \Phi_0 \\
&= (\sum c_i) + \Phi_N - \Phi_0
\end{aligned}$$

Da igualdade obtida

$$\sum_{i=1}^{N} \widehat{c_i} = (\sum_{i=1}^{N} c_i) + \Phi_N - \Phi_0$$

e das propriedades da função de potencial acima enumeradas, podemos concluir que

$$\sum_{i=1}^{N} \widehat{c_i} - (\sum_{i=1}^{N} c_i) = \Phi_N - \Phi_0 = \Phi_N - 0 \geq 0$$

E por isso, que a soma dos custos estimados $(\sum \widehat{c_i})$ é maior ou igual à soma dos custos reais $(\sum c_i)$.
(É de notar que uma formulação alternativa das propriedades da função de potencial poderia ser que o potencial de cada estado é sempre superior ao potencial do estado inicial.)

A definição da função de potencial nem sempre é muito fácil ou intuitiva. Esta função deve caracterizar o esforço envolvido em processar um dado estado. Daí que muitas vezes se associe o potencial de um estado com o seu estado de desordem.
No caso concreto que temos vindo a analisar, e depois de todo este foco nesse exemplo, a solução surge mais naturalmente.
Vamos definir o potencial de um array de bits como o número de `1`s desse array.
Note-se que esta função é independente da função `inc` que estamos a analisar. Depende apenas do valor do seu argumento (estado).
Definida esta função, calculemos o valor esperado do custo de cada operação. Este é dado, como foi dito acima, pela fórmula

$$\widehat{c_i} = c_i + \Phi_i - \Phi_{i-1}$$

É relativamente fácil, neste contexto, caracterizar o custo real de cada operação. De facto, o que a função `inc` está a fazer é a converter em `0`s o maior prefixo de `1`s do vector, seguido de converter um bit adicional.

- Assumindo que o array $b$ tem pelo menos um bit a `0`, seja `k` a posição do `0` mais à direita. Então, $\Phi(b) = l + r$ em que $l$ corresponde à soma do número de `1`s até à posição `k` e $r$ corresponde ao número de `1`s da posição `k` em diante (que não é mais do que o comprimento do maior prefixo de `1`s de $b$).

- No estado após a execução de `inc`, o potencial passa a ser $\Phi(b') = l + 1$ uma vez que os $r$ bits a `1` foram substituidos por `0`s.

O custo estimado de cada operação é então

$$\widehat{c_i} = (l + 1) + (r + 1) - (l + r) = 2$$

Obtendo (sem grande surpresa!) o mesmo resultado que obtivemos pela aplicação dos outros métodos.

# 4 Tabelas dinâmicas

Vamos terminar esta apresentação com o estudo de mais um caso em que a análise amortizada é usada.
Considere-se uma estrutura de dados implementada num array em que o custo de fazer uma inserçao é constante. Por simplicidade da apresentação, admitamos que esse custo é 1.
Vamos enriquecer essa implementação, permitindo que, quando queremos inserir um novo elmento e o array esgota a sua capacidade, (1) é realocado um novo array com o dobro da capacidade, (2) é copiado o conteúdo do array antigo para este novo array e, finalmente (3) é adicionado o novo elemento.
A análise do custo de cada inserção mostra-nos que

- no melhor caso (o array ainda não esgotou a sua capacidade) o custo da inserção é igual ao custo anterior, i.e., 1.

- no pior caso (a capacidade do array já foi atingida), temos um custo adicional de $N$, que corresponde à cópia dos elementos para o novo array.

Vejamos então a análise amortizada desta nova operação de inserção.

## 4.1 Análise Agregada

A tabela seguinte mostra o estado da tabela após a inserção das primeiras 12 entradas numa tabela inicialmente vazia (e com um tamanho inicial de 1), bem como os custos das várias inserções.

| i | Resultado | $c_i$ |
|---|---|---|
| 0 | | |
| 1 | 1 | 1 |
| 2 | 1 2 | 2 = 1+1 |
| 3 | 1 2 3 | 3 = 1+2 |
| 4 | 1 2 3 4 | 1 |
| 5 | 1 2 3 4 5 | 5 = 1+4 |
| 6 | 1 2 3 4 5 6 | 1 |
| 7 | 1 2 3 4 5 6 7 | 1 |
| 8 | 1 2 3 4 5 6 7 8 | 1 |
| 9 | 1 2 3 4 5 6 7 8 9 | 9 = 1+8 |
| 10 | 1 2 3 4 5 6 7 8 9 10 | 1 |
| 11 | 1 2 3 4 5 6 7 8 9 10 11 | 1 |
| 12 | 1 2 3 4 5 6 7 8 9 10 11 12 | 1 |

A inspecção dos vários custos associados a cada uma das operações põe em evidência que o somatório dos custos de uma sequência de $N$ inserções pode ser calculado como

$$C_N = N + \sum_{i=0}^{?} 2^i$$

Mais uma vez, o limite superior deste somatório corresponde ao número de vezes que conseguimos dividir $N$ por 2 (sem chegar a 0), i.e., $\log_2 N$.
Desenvolvendo,

$$C_N = N + \sum_{i=0}^{\log_2 N} 2^i = N + (2^{1+\log_2 N} - 1) = N + 2\,N - 1 = 3\,N - 1$$

Do que resulta um custo amortizado $\widehat{c}_i$ de

$$\widehat{c}_i = \frac{1}{N}\,C_N = \frac{1}{N}\,(3\,N - 1) = 3 - \frac{1}{N} = \Theta(1)$$

## 4.2 Método Contabilístico

Admitindo que o saldo mínimo é atingido depois de uma duplicação da tabela, cada nova inserção deverá ter como custo amortizado, para além do custo unitário de uma inserção, um custo suplementar de 2, que será usado para a próxima duplicação da tabela. Isto porque se a tabela passar a ter um tamanho $X$, a nova duplicação acontecerá após $X/2$ inserções. Por isso cada inserção deverá *amealhar* 2.
Vejamos então a tabela da análise anterior, aumentada com o custo estimado $\widehat{c}_i = 3$ e o saldo no fim de cada operação.

| i | Resultado | $c_i$ | $\widehat{c}_i$ | $B_i$ |
|---|---|---|---|---|
| 0 | | | | 0 |
| 1 | 1 | 1 | 3 | 2 |
| 2 | 1 2 | 2 | 3 | 3 |
| 3 | 1 2 3 | 3 | 3 | 3 |
| 4 | 1 2 3 4 | 1 | 3 | 5 |
| 5 | 1 2 3 4 5 | 5 | 3 | 3 |
| 6 | 1 2 3 4 5 6 | 1 | 3 | 5 |
| 7 | 1 2 3 4 5 6 7 | 1 | 3 | 7 |
| 8 | 1 2 3 4 5 6 7 8 | 1 | 3 | 9 |
| 9 | 1 2 3 4 5 6 7 8 9 | 9 | 3 | 3 |
| 10 | 1 2 3 4 5 6 7 8 9 10 | 1 | 3 | 5 |
| 11 | 1 2 3 4 5 6 7 8 9 10 11 | 1 | 3 | 7 |
| 12 | 1 2 3 4 5 6 7 8 9 10 11 12 | 1 | 3 | 9 |

### 4.3 Método do Potencial

De forma a definirmos o valor do potencial de cada estado, devemos atentar no seguinte:

- O potencial deverá aumentar à medida que o número de elementos da tabela aumenta.
- Após uma operação que envolva a duplicação da tabela, o potencial deve diminuir proporcionalmente ao número de elementos na tabela (de forma a absorver o custo adicional de cópia dos elementos).

Uma forma de obedecer a estes dois requisitos será definir o potencial de um estado $i$ como

$$\Phi_i = U_i - F_i = 2 * U_i - S_i$$

em que $S_i$, $U_i$ e $F_i$ correspondem ao tamanho do array, ao número de posições ocupadas e número de posições livres respectivamente.
De forma a garantir uma das propriedades da função de potencial, diremos ainda que $\Phi_0 = 0$. A segunda propriedade, que o potencial nunca se torna negativo, é um invariante da estrutura em causa: quando se duplica o tamanho, pelo menos metade estará ocupada.
Vejamos então qual o valor do custo estimado de cada inserção. Para isso analisaremos os dois comportamentos alternativos da função.

- No caso de **não se fazer a duplicação** da tabela: o custo real da operação é $c_i = 1$; o tamanho do array permanece inalterado ($S_i = S_{i-1}$); o número de posições livres diminui uma unidade ($F_i = F_{i-1} - 1$); o número de posições ocupadas aumenta uma unidade ($U_i = U_{i-1} + 1$).

$$\begin{aligned}
\widehat{c_i} &= c_i + (\Phi_i - \Phi_{i-1}) \\
&= 1 + (U_i - F_i) - (U_{i-1} - F_{i-1}) \\
&= 1 + (U_{i-1} + 1) - (F_{i-1} - 1) - (U_{i-1} - F_{i-1}) \\
&= 1 + U_{i-1} + 1 - F_{i-1} + 1 - U_{i-1} + F_{i-1} \\
&= 3
\end{aligned}$$

- No caso de **se fazer a duplicação** da tabela: o custo real da operação é $c_i = 1 + U_{i-1}$ (note-se que $U_k$ corresponde ao número de posições usadas após a operação $k$); o tamanho do array aumenta par o dobro ($S_i = 2 * S_{i-1}$); o número de células usadas aumenta uma unidade ($U_i = U_{i-1} + 1$); o número de células livres aumenta de $F_{i-1} = 0$ para $F_i = U_{i-1} - 1$.

$$\begin{aligned}
\widehat{c}_i &= c_i + (\Phi_i - \Phi_{i-1}) \\
&= 1 + U_{i-1} + (U_i - F_i) - (U_{i-1} - F_{i-1}) \\
&= 1 + U_{i-1} + (U_{i-1} + 1 - (U_{i-1} - 1)) - (U_{i-1} - 0) \\
&= 1 + U_{i-1} + U_{i-1} + 1 - U_{i-1} + 1 - U_{i-1} + 0 \\
&= 1 + U_{i-1} + U_{i-1} + 1 - U_{i-1} + 1 - U_{i-1} + 0 \\
&= 3
\end{aligned}$$

Vemos então que, para ambos os casos, $\widehat{c}_i = 3$.

## 4.4 Remoção

Consideremos agora que adicionalmente à operação de inserção definida atrás, pretendemos implementar um procedimento de remoção que, no caso da tabela atingir um determinado valor mínimo da sua taxa de ocupação, ela é realocada para uma tabela menor.
Vamos realocar a tabela para metade do seu valor apenas quando a tabela tiver apenas um quarto das suas entradas preenchidas. A tabela seguinte parte duma tabela em que foram feitas 9 inserções e onde se vão sucessivamente removendo elementos.

| **i** | **Resultado** | **$c_i$** |
|---|---|---|
| 0 | 1 2 3 4 5 6 7 8 9 \| \| \| \| \| \| \| | |
| 1 | 1 2 3 4 5 6 7 8 \| \| \| \| \| \| \| \| | 1 |
| 2 | 1 2 3 4 5 6 7 \| \| \| \| \| \| \| \| \| | 1 |
| 3 | 1 2 3 4 5 6 \| \| \| \| \| \| \| \| \| \| | 1 |
| 4 | 1 2 3 4 5 \| \| \| \| \| \| \| \| \| \| \| | 1 |
| 5 | 1 2 3 4 \| \| \| \| | 5 |
| 6 | 1 2 3 \| \| \| \| \| | 1 |
| 7 | 1 2 \| \| | 3 |
| 8 | 1 \| | 1 |

No modelo contabilístico podemos associar a cada remoção um custo estimado de 2, correspondendo ao custo real 1 acrescido de uma *poupança* de 1 para a futura cópia: note-se que só após removermos (pelo menos) $N$ elementos é que teremos que copiar outros tantos $N$ elementos para a tabela menor.
O caso em que a remoção é *cara* corresponde a uma diminuição do tamanho da tabela. E nesse caso devemos ter *amealhado* o suficiente para cobrir esses gastos.

| i | Resultado | | | | | | | | | | | | | | | | $c_i$ | $\widehat{c}_i$ | $B_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | | | | | | | | | 0 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | | | | | | | | 1 | 2 | 1 |
| 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | | | | | | | | 1 | 2 | 2 |
| 3 | 1 | 2 | 3 | 4 | 5 | 6 | | | | | | | | | | | 1 | 2 | 3 |
| 4 | 1 | 2 | 3 | 4 | 5 | | | | | | | | | | | | 1 | 2 | 4 |
| 5 | 1 | 2 | 3 | 4 | | | | | | | | | | | | | 5 | 2 | 1 |
| 6 | 1 | 2 | 3 | | | | | | | | | | | | | | 1 | 2 | 2 |
| 7 | 1 | 2 | | | | | | | | | | | | | | | 3 | 2 | 1 |
| 8 | 1 | | | | | | | | | | | | | | | | 1 | 2 | 2 |

Em termos de função de potencial, o potencial deve aumentar à medida que a tabela se vai esvaziando (o número de células livres aumenta) máximo deve ser atingido quando a tabela fica apenas com 25% da sua capacidade preenchida, baixando nessa altura para o seu valor mínimo com a taxa de ocupação a 50%.
Seja então a função de potencial definida como

$$\Phi_i = \max(0, F_i - \frac{S_i}{2})$$

em que $F$ e $S$ representam o número de posições na tabela e o número de posições desocupadas (a expressão $\max(0, \_)$ garante apenas que o potencial se mantem não negativo).
Analizemos então os custos estimados face a esta definição

- No caso em que o número de posições desocupadas se mantem a menos de metade da tabela, o potencial mantém-se a zero.

$$\widehat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + 0 - 0 = 1$$

- Nos restantes casos em que a dimensão da tabela não é alterada ($S_i = S_{i-1}$)

$$\begin{aligned}\widehat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (F_{i-1} + 1) - \tfrac{S_i}{2} - (F_{i-1} - \tfrac{S_{i-1}}{2}) \\ &= 1 + F_{i-1} + 1 - \tfrac{S_i}{2} - F_{i-1} + \tfrac{S_i}{2} \\ &= 2\end{aligned}$$

- Finalmente, quando a tabela é redimensionada para metade ($S_i = \frac{S_{i-1}}{2}$), o número de posições livres passa a ser metade do tamanho da tabela $F_i = \frac{S_i}{2} = \frac{S_{i-1}}{4}$. Nesse caso, o custo real $c_i$ contem também o custo de copiar tantos elementos quanto as posições que ficaram desocupadas ($c_i = 1 + F_i$).

  Note-se que para que a tabela seja realocada é necessário que após a remoção, 3/4 da tabela estejam desocupados, i.e., $F_{i-1} = \frac{3}{4}\, S_{i-1} - 1 = \frac{3}{2}\, S_i - 1 = 3\, F_i - 1$

$$\begin{aligned}&\widehat{c}_i = c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + F_i + (F_i - \tfrac{S_i}{2}) - (F_{i-1} - \tfrac{S_{i-1}}{2}) \\ &= 1 + F_i + F_i - \tfrac{S_i}{2} - F_{i-1} + \tfrac{S_{i-1}}{2} \\ &= 1 + F_i + F_i - F_i - (3\, F_i - 1) + 2\, F_i \\ &= 2\end{aligned}$$