

Análise de Algoritmos 1º/2020

Izabela Ramos Ferreira – 2012130024

Tabela Dinâmica

Código [aqui](#)

- a) calcular o tempo médio (5 execuções) de 100.000.000 de inserções e 100.000.000 de remoções em um vetor de tamanho fixo de 100.000.000

```
izabela@izabela-pc: ~/Documentos/analise_algoritmos/tabela_dinamica
Arquivo Editar Ver Pesquisar Terminal Ajuda
izabela@izabela-pc:~/Documentos/analise_algoritmos/tabela_dinamica
$ gcc tabeladinamica.c -o tabeladinamica
izabela@izabela-pc:~/Documentos/analise_algoritmos/tabela_dinamica
$ ./tabeladinamica

Tempo em segundos: 1.334465
izabela@izabela-pc:~/Documentos/analise_algoritmos/tabela_dinamica
$ █
```

- b) calcular o tempo médio (5 execuções) de 100.000.000 de inserções e 100.000.000 de remoções em um vetor dinâmico. Com aumento * 2 e diminuição /2.

```
izabela@izabela-pc: ~/Documentos/analise_algoritmos/tabela_dinamica
Arquivo Editar Ver Pesquisar Terminal Ajuda
izabela@izabela-pc:~/Documentos/analise_algoritmos/tabela_dinamica
$ gcc tabeladinamica.c -o tabeladinamica
izabela@izabela-pc:~/Documentos/analise_algoritmos/tabela_dinamica
$ ./tabeladinamica

Tempo em segundos: 1.193302
izabela@izabela-pc:~/Documentos/analise_algoritmos/tabela_dinamica
$ █
```

- c) calcular o tempo médio (5 execuções) de 100.000.000 de inserções e 100.000.000 de remoções em um vetor dinâmico. Com aumento * 3 e diminuição /3.

```
izabela@izabela-pc: ~/Documentos/analise_algoritmos/tabela_dinamica
Arquivo Editar Ver Pesquisar Terminal Ajuda
izabela@izabela-pc:~/Documentos/analise_algoritmos/tabela_dinamica
$ gcc tabeladinamica.c -o tabeladinamica
izabela@izabela-pc:~/Documentos/analise_algoritmos/tabela_dinamica
$ ./tabeladinamica

Tempo em segundos: 1.239551
izabela@izabela-pc:~/Documentos/analise_algoritmos/tabela_dinamica
$ █
```

d) Qual a complexidade de tempo das alternativas A, B e C.

O(n)

e) Dobrar ao invés de triplicar o tamanho na tabela dinâmica é pior ou melhor?

Em qual sentido? Justifique.

Dobrar é melhor porque depois de uma diminuição da tabela a carga deve dobrar antes que um aumento seja necessário e assim o número de elementos inseridos será o limite necessário para a expansão da tabela.

Porém dobrar o tamanho, depois de uma expansão, exige que a carga seja reduzida à metade antes que ocorra uma diminuição da tabela para que o número de elementos removidos seja o limite necessário para a diminuição da tabela.

Matematicamente, considerando c a carga da tabela e t o tamanho da tabela, precisa de pelo menos $2*c - t$ se $c \geq t/2$ e pelo menos $t/2 - c$ se $c < t/2$.

f) faça a dedução matemática da complexidade da utilização da tabela dinâmica.

A tabela dinâmica com inserção e deleção tem complexidade O(n) e O(1) com amortização (Uma análise do comportamento geral do algoritmo avaliando o resultado sem a influência das iterações anteriores):

O(n) porque em 1 execução o custo mínimo de tempo é pelo menos $1*(1*tamanho)$. Na seguinte, será $n*(1*tamanho + 2*tamanho)$, sendo n a representação do ciclo dessa iteração. E por aí vai... Computacionalmente só importa o n , portanto: O(n).

O(1) porque a amortização da análise evidencia o fluxo do algoritmo fora dos ciclos de iteração (remove o "n"), restando portanto: $(1*tamanho)$ que representa um valor que em questão de grandeza depende de computador para computador, mas que em questão de análise equivale simbolicamente a 1 unidade de custo.

Cada execução consome pelo menos 1 unidade de custo no melhor caso e consome : **carga +1** em unidades de custo no pior caso. Se o tamanho = 0 então a carga = 0 e a expressão matemática que representa essa situação é $2*carga - tamanho = 0$. No fim da execução do algoritmo o "esperado" é carga = tamanho = 1, aonde $2*carga - tamanho = 1$, considerando c a carga da tabela e t o tamanho da tabela:

$$2*c - t = 1, \text{ se } c = t = 1:$$

$$2*1 - 1 = 1.$$

$$\therefore O(1)$$

E a expressão $2*carga - tamanho$ acontece, mas sempre dentro de um laço n:

$$\therefore n*(2*c - t) \rightarrow O(n)$$