Algoritmos

Pedro Hokama

Fontes

- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden
- Desmistificando Algoritmos, Thomas H. Cormen.
- Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Stanford Algorithms
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V`
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6`
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende
- Conjunto de Slides do Professores Cid C. de Souza para a disciplina MO420

Qualquer erro é de minha responsabilidade.

Análise - QuickSort

Teorema

Para qualquer entrada de comprimento n, o tempo esperado de execução do QuickSort (com pivôs aleatórios) é $O(n \log n)$.

- Primeiramente considere como entrada um arranjo A de comprimento n.
- Espaço amostral: Ω = todos as possíveis sequências de escolhas de pivôs no QuickSort.
- Variável aleatória: para qualquer $\sigma \in \Omega$, $C(\sigma)$ = número de comparações entre dois elementos do arranjo feito pelo algoritmo QuickSort dado as escolhas σ. Note que o tempo de execução total do QuickSort é dominado por esse número.
- O objetivo então é mostrar que $E[C] = O(n \log n)$

- Notação:
  - z_i = i-ésimo menor elemento
  - ou seja, z_i não é o elemento que está originalmente na i-ésima posição do vetor, mas sim o elemento que vai ocupar a i-ésima posição no vetor quando ordenado.

$$(\underbrace{6}_{z_2}, \underbrace{10}_{z_4}, \underbrace{8}_{z_3}, \underbrace{2}_{z_1})$$

  - para uma escolha σ, e $i < j$, seja $X_{ij}(\sigma)$ = número de vezes que z_i e z_j são comparados.
- Fixado dois elementos da entrada, quantas vezes eles podem ser comparados?
  a. 1
  b. 0 ou 1
  c. 0, 1 ou 2
  d. algo entre 0 e $n-1$

Algoritmo 1: QuickSort

Entrada: Um arranjo A, índices l e r

Saída: Um arranjo com os mesmos elementos de A porém ordenados

```
1 se r − l ≤ 0 então devolva A;
2 p = Partição(A, l, r);
3 QuickSort(A, l, p-1);
4 QuickSort(A, p+1, r);
5 devolva A;
```

Algoritmo 2: Partição

Entrada: Um arranjo A, índices l e r

Saída: O mesmo arranjo mas particionado

```
1 Coloca o pivot em A[l] ;
2 pivot = A[l] ;
3 i = l + 1;
4 para j = l + 1 até r faça
5     se A[j] < pivot então
6         Troca A[j] e A[i];
7         i = i + 1;
8 Troca A[l] e A[i − 1];
9 devolva i − 1;
```


- Podemos então facilmente escrever C em função de X

$$C(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}(\sigma), \qquad \forall \sigma \in \Omega$$

- Pela linearidade da esperança (lembrando que ela se aplica mesmo se as variáveis não forem independentes)

$$E[C] = E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}].$$

- Como:

$$\begin{aligned} E[X_{ij}] &= 0 \cdot Pr[X_{ij} = 0] + 1 \cdot Pr[X_{ij} = 1] \\ &= \cancel{0 \cdot Pr[X_{ij} = 0]} + 1 \cdot Pr[X_{ij} = 1] \\ &= Pr[X_{ij} = 1] = Pr[z_i \text{ e } z_j \text{ serem comparados}] \end{aligned}$$

$$E[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i \text{ e } z_j \text{ serem comparados}]$$

Lema

Para todo $i < j$, $Pr[z_i \text{ e } z_j \text{ serem comparados}] = \frac{2}{(j-i+1)}$

- Fixe z_i, z_j com $i < j$.
- Considere o conjunto $\{z_i, z_{i+1}, \ldots, z_{j-1}, z_j\}$.
- Desde que nenhum desses números seja escolhido como pivô, todos serão passados juntos para a mesma chamada recursiva.
- Considere então o primeiro entres $\{z_i, z_{i+1}, \ldots, z_{j-1}, z_j\}$ que é escolhido como pivô:
  - se z_i ou z_j for escolhido primeiro, eles serão escolhidos.
  - se escolher um dos outros então z_i e z_j nunca serão comparados.
- como a escolha é aleatória qualquer um dos elementos do conjunto tem a mesma chance de ser escolhido primeiro (do conjunto) e portando a chance deles serem comparados é

$$2/(j - i + 1)$$

□

Então

$$E[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i \text{ e } z_j \text{ serem comparados}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{(j - i + 1)}$$

$$E[C] = 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{(j - i + 1)}$$

.

- Agora note que a soma interna

$$\sum_{j=i+1}^{n} \frac{1}{(j - i + 1)} = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n - i + 1}$$

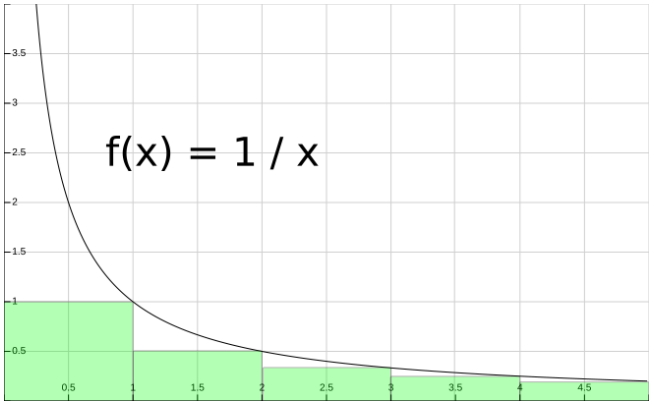
- e ela obtêm a maior quantidade de termos (e o maior valor) quando $i = 1$
- A maior soma então é

$$\sum_{j=2}^{n} \frac{1}{j} = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n}$$

Limitamos então superiormente a esperança por:

$$\begin{aligned} E[C] &= 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{(j-i+1)} \\ &\leq 2\cdot n\cdot\sum_{j=i+1}^{n}\frac{1}{(j-i+1)} \\ &\leq 2\cdot n\cdot\sum_{k=2}^{n}\frac{1}{k} \end{aligned}$$

$$E[C] \leq 2\cdot n\cdot\sum_{k=2}^{n}\frac{1}{k}$$



A soma agora pode ser limitada superiormente por:

$$\sum_{k=2}^{n}\frac{1}{k} \leq \int_{1}^{n}\frac{1}{x}dx = \ln x\Big|_{1}^{n} = \ln n - \cancel{\ln 1} = \ln n$$

.
Portanto:

$$E[C] \leq 2\cdot n\cdot \ln n$$

e portanto:
O tempo de execução esperado do QuickSort é $O(n\log n)$ □