Análise Probabilística e Algoritmos Randomizados

Prof. Luiz Chaimowicz

Análise Probabilística

- Consiste no uso de probabilidades na análise do tempo de execução de algoritmos, principalmente no caso médio
- Considera que a variação dos dados de entrada segue alguma distribuição de probabilidades e computa o tempo médio de execução sobre todas as entradas

Exemplo: *the hiring problem*

- Queremos contratar um novo funcionário para uma posição estratégica da empresa e para isso usamos uma agência de colocação
- A agência manda funcionários para entrevista. A cada uma, pagamos um valor c_i para a agência
- Para cada funcionário contratado, temos custos envolvidos com a demissão e contratação: c_h
  - $c_h > c_i$
- Apesar disso, a nossa estratégia é sempre estar com o melhor funcionário para a posição

```
HIRE-ASSISTANT(n)
best = 0          // candidate 0 is a least-qualified dummy candidate
for i = 1 to n
    interview candidate i        c_i
    if candidate i is better than candidate best
        best = i
        hire candidate i         c_h
```

Qual o "custo financeiro" desse algoritmo?

$O(c_i n + c_h m)$, para *m* contratações, onde *m* varia com a ordem das entrevistas

Considerando apenas o custo das contratações:

Pior caso: $O(c_h n)$ –> Contrata todos, do pior ao melhor

Melhor caso: $O(c_h)$ –> O primeiro é o melhor

Análise Probabilística

E na média, qual o custo de contrataçao?

- Para a análise probabilística, vamos considerar que a distribuição dos candidatos é aleatória
- Mais formalmente:
  - Existe uma ordenação da qualidade dos funcionários *rank(1), rank(2), ..., rank(n)*
  - A entrada <1, 2, ..., n> é uma permutação dessa lista ou seja, um sorteio de 1 em n! possibilidades
  - *Uniform Random Permutation*

Probabilidades

(Revisão de Probabilidades: ver apêndice C)

Informalmente, dado um **espaço amostral *S*,** a **probabilidade** nos fornece a **"chance" de um evento *A* acontecer** onde *A* é subconjunto de *S*

Exemplo: Cara (H) ou Coroa (T)

$S = \{H, T\}$, $\Pr\{H\} = ½$, $\Pr\{T\}=½$

Uma **variável aleatória** é uma função que associa eventos de um espaço amostral *S* a números reais x. X(s) = x

Probabilidades

Exemplo: considere jogar 2 dados distintos. Existem 36 eventos e a probabilidade de cada um é Pr{s} = 1/36.

Vamos definir uma variável aleatória *X* como o maior dos dois valores obtidos nos dados.

Qual é Pr{*X*=3}?

Pr{X=3} = 5/36

Eventos: (1,3), (2,3), (3,3), (3,2), (3,1)

Probabilidades

O **valor esperado** de uma variável aleatória é dado por: $E[X] = \sum_x x.\Pr\{X = x\}$

Exemplo: suponha um jogo de cara ou coroa onde uma moeda é jogada duas vezes. Ganhamos $3 para cada cara (H) e perdemos $2 a cada coroa (T).

Qual o valor esperado de uma variável aleatória X que representa o nosso lucro?

E[x]= 6.Pr{2H} + 1.Pr{1H,1T} – 4.Pr{2T}

= 6.(1/4) + 1.(1/2) – 4.(1/4) = 1

Variável Aleatória Indicadora

- Fornece uma forma conveniente de converter entre probabilidades e valores esperados.
- Uma variável aleatória indicadora I de um evento *A* é definida como:

$$I\{A\} = \begin{cases} 1, \text{ se A ocorrer} \\ 0, \text{ se A não ocorrer} \end{cases}$$

***Lema 5.1*: Dado um espaço amostral *S* e um evento *A*, seja $X_a = I\{A\}$. Então $E[X_a] = Pr\{A\}$**

Análise: *the hiring problem*

- Considerando que os funcionários chegam em ordem aleatória para entrevista, qual o número esperado de vezes que contratamos um novo funcionário?
  - #vezes que as linhas 5 e 6 são executadas
- Seja *X* uma variável aleatória que representa o número de vezes que contratamos um funcionário.O valor esperado de *X* é dado por:

$$E[X] = \sum_{1}^{n} x.\Pr\{X = x\}$$

Análise: *the hiring problem*

- Para simplificar os cálculos, vamos considerar *n* variáveis aleatórias indicadoras representando a contratação de cada candidato *i*

$$X_i = I\{\text{candidato } i \text{ é contratado}\} = \begin{cases} 1, \text{ se } i \text{ é contratado} \\ 0, \text{ se } i \text{ não é contratado} \end{cases}$$

Logo: $X = X_1 + X_2 + ... + X_n$

Análise: *the hiring problem*

- Pelo lema 5.1, temos que:
  $E[X_i]$ = Pr{X_i} = Pr{candidato *i* é contratado}
- Um **candidato *i* é contratado quando** ele é melhor que todos os *i-1* que vieram antes dele, ou seja, **ele é o melhor dentre *i* candidatos**
- Como a ordem de chegada é aleatória, **a chance dele ser o melhor dentre os *i* é *1/i***

Logo: $E[X_i]$ = Pr{candidato *i* é contratado} = *1/i*

Análise: *the hiring problem*

Resolvendo E[*X*] com um pouco de matemática...

$$
\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] && \text{(by equation (5.2))} \\
&= \sum_{i=1}^{n} \mathrm{E}[X_i] && \text{(by linearity of expectation)} \\
&= \sum_{i=1}^{n} 1/i && \text{(by equation (5.3))} \\
&= \ln n + O(1) && \text{(by equation (A.7))} .
\end{aligned}
$$

Análise: *the hiring problem*

```
HIRE-ASSISTANT(n)
best = 0        // candidate 0 is a least-qualified dummy candidate
for i = 1 to n
    interview candidate i
    if candidate i is better than candidate best
        best = i
        hire candidate i
```

Portanto, no caso médio, o custo das contratações é: ***O($c_h$ln(n))*** *, que é muito melhor que o pior caso*

Requer que a distribuição inicial dos candidatos seja aleatória

Algoritmos Randomizados

- O que acontece se não for possível conhecer a distribuição dos dados de entrada?
  - Podemos impor uma distribuição!

Algoritmos Randomizados:

De forma geral, um algoritmo randomizado tem o seu comportamento definido não só pela entrada mas pelo uso de um **gerador de números aleatórios**

Algoritmos Randomizados

No caso do *hiring problem*, podemos garantir a distribuição fazendo uma permutação aleatória do vetor de entrada no início

```
RANDOMIZED-HIRE-ASSISTANT(n)
randomly permute the list of candidates
best = 0        // candidate 0 is a least-qualified dummy candidate
for i = 1 to n
    interview candidate i
    if candidate i is better than candidate best
        best = i
        hire candidate i
```

Algoritmos Randomizados

- Em um algoritmo determinístico, para uma mesma entrada é sempre produzida a mesma saída
- No algoritmo randomizado, isso não ocorre
  - Pode ser interessante para "fugir" do pior caso
  - O algoritmo funciona sempre no caso médio
- Podemos aplicar as técnicas de análise probabilística sem a necessidade de se conhecer a distribuição da entrada
- Possível problema: custo para randomizar

Custo para permutar um vetor

```
PERMUTE-BY-SORTING(A)
n = A.length
let P[1..n] be a new array
for i = 1 to n
    P[i] = RANDOM(1, n^3)
sort A, using P as sort keys
```

Ω(nlgn)

```
RANDOMIZE-IN-PLACE(A)
n = A.length
for i = 1 to n
    swap A[i] with A[RANDOM(i, n)]
```

Θ(n)

Prova de que ambos funcionam no livro...

Para Casa

- Ler Capítulo 5 do Cormen
- Fazer outros exercícios: ex. 5.3.4