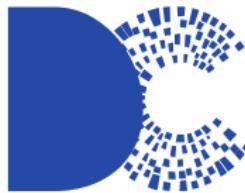




UNIVERSIDADE
FEDERAL DE
SERGIPE



DEPARTAMENTO
DE COMPUTAÇÃO

Algoritmos probabilísticos

Projeto e Análise de Algoritmos

Bruno Prado

Departamento de Computação / UFS

Introdução

- ▶ Teoria da probabilidade
 - ▶ É utilizada para calcular as chances de ocorrência de um determinado evento possível em um espaço S

Introdução

- ▶ Teoria da probabilidade
 - ▶ É utilizada para calcular as chances de ocorrência de um determinado evento possível em um espaço S
 - ▶ A probabilidade de ocorrência de um evento A é definida por $0 \leq P[A] \leq P[S] = 1$, onde $P[S]$ é a probabilidade de qualquer evento possível acontecer

Introdução

- ▶ Teoria da probabilidade
 - ▶ É utilizada para calcular as chances de ocorrência de um determinado evento possível em um espaço S
 - ▶ A probabilidade de ocorrência de um evento A é definida por $0 \leq P[A] \leq P[S] = 1$, onde $P[S]$ é a probabilidade de qualquer evento possível acontecer
 - ▶ As chances de um evento qualquer que não seja A acontecer é $P[\neg A] = P[S] - P[A] = 1 - P[A]$

Introdução

► Teoria da probabilidade

- É utilizada para calcular as chances de ocorrência de um determinado evento possível em um espaço S
 - A probabilidade de ocorrência de um evento A é definida por $0 \leq P[A] \leq P[S] = 1$, onde $P[S]$ é a probabilidade de qualquer evento possível acontecer
 - As chances de um evento qualquer que não seja A acontecer é $P[\neg A] = P[S] - P[A] = 1 - P[A]$
 - A ocorrência dos eventos independentes A e B tem probabilidade $P[A \cap B] = P[A] \times P[B]$

Introdução

► Teoria da probabilidade

- É utilizada para calcular as chances de ocorrência de um determinado evento possível em um espaço S
 - A probabilidade de ocorrência de um evento A é definida por $0 \leq P[A] \leq P[S] = 1$, onde $P[S]$ é a probabilidade de qualquer evento possível acontecer
 - As chances de um evento qualquer que não seja A acontecer é $P[\neg A] = P[S] - P[A] = 1 - P[A]$
 - A ocorrência dos eventos independentes A e B tem probabilidade $P[A \cap B] = P[A] \times P[B]$
 - Para que os eventos A ou B aconteçam, as chances são $P[A \cup B] = P[A] + P[B] - P[A \cap B]$

Introdução

► Teoria da probabilidade

- É utilizada para calcular as chances de ocorrência de um determinado evento possível em um espaço S
 - A probabilidade de ocorrência de um evento A é definida por $0 \leq P[A] \leq P[S] = 1$, onde $P[S]$ é a probabilidade de qualquer evento possível acontecer
 - As chances de um evento qualquer que não seja A acontecer é $P[\neg A] = P[S] - P[A] = 1 - P[A]$
 - A ocorrência dos eventos independentes A e B tem probabilidade $P[A \cap B] = P[A] \times P[B]$
 - Para que os eventos A ou B aconteçam, as chances são $P[A \cup B] = P[A] + P[B] - P[A \cap B]$
 - Um evento impossível ϕ tem probabilidade $P[\phi] = 0$

Introdução

- ▶ Teoria da probabilidade
 - ▶ Considerando dados honestos com 6 faces, existe um espaço de eventos possíveis $S = \{1, 2, 3, 4, 5, 6\}$

Introdução

- ▶ Teoria da probabilidade
 - ▶ Considerando dados honestos com 6 faces, existe um espaço de eventos possíveis $S = \{1, 2, 3, 4, 5, 6\}$
 - ▶ Qual é a probabilidade dos eventos independentes A (exceto o número 1) e B (qualquer número par)?

Introdução

- ▶ Teoria da probabilidade
 - ▶ Considerando dados honestos com 6 faces, existe um espaço de eventos possíveis $S = \{1, 2, 3, 4, 5, 6\}$
 - ▶ Qual é a probabilidade dos eventos independentes A (exceto o número 1) e B (qualquer número par)?

$$\begin{aligned}P[A] &= 1 - P[1] \\&= 1 - \frac{1}{6} \\&= \frac{5}{6} \\&\approx 83\%\end{aligned}$$

Introdução

- ▶ Teoria da probabilidade
 - ▶ Considerando dados honestos com 6 faces, existe um espaço de eventos possíveis $S = \{1, 2, 3, 4, 5, 6\}$
 - ▶ Qual é a probabilidade dos eventos independentes A (exceto o número 1) e B (qualquer número par)?

$$\begin{aligned}P[A] &= 1 - P[1] \\&= 1 - \frac{1}{6} \\&= \frac{5}{6} \\&\approx 83\%\end{aligned}$$

$$\begin{aligned}P[B] &= P[2] + P[4] + P[6] \\&= \frac{1}{6} + \frac{1}{6} + \frac{1}{6} \\&= \frac{1}{2} \\&= 50\%\end{aligned}$$

Introdução

- ▶ Teoria da probabilidade
 - ▶ Considerando dados honestos com 6 faces, existe um espaço de eventos possíveis $S = \{1, 2, 3, 4, 5, 6\}$
 - ▶ Qual é a probabilidade dos eventos independentes C (evento A ou B) e D (três eventos A seguidos)

Introdução

- ▶ Teoria da probabilidade
 - ▶ Considerando dados honestos com 6 faces, existe um espaço de eventos possíveis $S = \{1, 2, 3, 4, 5, 6\}$
 - ▶ Qual é a probabilidade dos eventos independentes C (evento A ou B) e D (três eventos A seguidos)

$$\begin{aligned}P[C] &= P[A \cup B] \\&= P[A] + P[B] - P[A \cap B] \\&= \frac{5}{6} + \frac{1}{2} - \left(\frac{5}{6} \times \frac{1}{2}\right) \\&= \frac{11}{12} \\&\approx 92\%\end{aligned}$$

Introdução

► Teoria da probabilidade

- Considerando dados honestos com 6 faces, existe um espaço de eventos possíveis $S = \{1, 2, 3, 4, 5, 6\}$
- Qual é a probabilidade dos eventos independentes C (evento A ou B) e D (três eventos A seguidos)

$$P[C] = P[A \cup B]$$

$$= P[A] + P[B] - P[A \cap B]$$

$$= \frac{5}{6} + \frac{1}{2} - \left(\frac{5}{6} \times \frac{1}{2} \right)$$

$$= \frac{11}{12}$$

$$\approx 92\%$$

$$P[D] = P[A \cap A \cap A]$$

$$= P[A] \times P[A] \times P[A]$$

$$= \frac{5}{6} \times \frac{5}{6} \times \frac{5}{6}$$

$$= \frac{125}{216}$$

$$\approx 58\%$$

Introdução

► Teoria da probabilidade

- Caso os eventos sejam dependentes, ou seja, a ocorrência de um determinado evento B influenciar as chances de outro evento A acontecer ($P[A|B]$) ou do evento A afetar as probabilidades do evento B ocorrer ($P[B|A]$), é utilizada a probabilidade condicional ou teorema de Bayes

| | | β | $\neg\beta$ | |
|---|----------|---------|-------------|-----------------|
| | | R | S | $R + S$ |
| A | A | R | S | $R + S$ |
| | $\neg A$ | U | V | $U + V$ |
| | | $R + U$ | $S + V$ | $R + S + U + V$ |

Introdução

► Teoria da probabilidade

- Caso os eventos sejam dependentes, ou seja, a ocorrência de um determinado evento B influenciar as chances de outro evento A acontecer ($P[A|B]$) ou do evento A afetar as probabilidades do evento B ocorrer ($P[B|A]$), é utilizada a probabilidade condicional ou teorema de Bayes

| | | β | $\neg\beta$ | |
|---|----------|---------|-------------|-----------------|
| | | R | S | $R + S$ |
| A | $\neg A$ | U | V | $U + V$ |
| | | $R + U$ | $S + V$ | $R + S + U + V$ |

$$P[A] = \frac{r+s}{r+s+u+v}$$

Introdução

► Teoria da probabilidade

- Caso os eventos sejam dependentes, ou seja, a ocorrência de um determinado evento B influenciar as chances de outro evento A acontecer ($P[A|B]$) ou do evento A afetar as probabilidades do evento B ocorrer ($P[B|A]$), é utilizada a probabilidade condicional ou teorema de Bayes

| | | β | $\neg\beta$ | |
|---|----------|---------|-------------|-----------------|
| | | R | S | $R + S$ |
| A | A | R | S | $R + S$ |
| | $\neg A$ | U | V | $U + V$ |
| | | $R + U$ | $S + V$ | $R + S + U + V$ |

$$P[B] = \frac{r+u}{r+s+u+v}$$

Introdução

► Teoria da probabilidade

- Caso os eventos sejam dependentes, ou seja, a ocorrência de um determinado evento B influenciar as chances de outro evento A acontecer ($P[A|B]$) ou do evento A afetar as probabilidades do evento B ocorrer ($P[B|A]$), é utilizada a probabilidade condicional ou teorema de Bayes

| | β | $\neg\beta$ | |
|----------|---------|-------------|-----------------|
| A | R | S | $R + S$ |
| $\neg A$ | U | V | $U + V$ |
| | $R + U$ | $S + V$ | $R + S + U + V$ |

$$P[A|B] = \frac{r}{r+u} = \frac{P[B|A] \times P[A]}{P[B]}$$

Introdução

- ▶ Teoria da probabilidade
 - ▶ Caso os eventos sejam dependentes, ou seja, a ocorrência de um determinado evento B influenciar as chances de outro evento A acontecer ($P[A|B]$) ou do evento A afetar as probabilidades do evento B ocorrer ($P[B|A]$), é utilizada a probabilidade condicional ou teorema de Bayes

| | β | $\neg\beta$ | |
|----------|---------|-------------|-----------------|
| A | R | S | $R + S$ |
| $\neg A$ | U | V | $U + V$ |
| | $R + U$ | $S + V$ | $R + S + U + V$ |

$$P[B|A] = \frac{r}{r+s} = \frac{P[A|B] \times P[B]}{P[A]}$$

Introdução

- ▶ Teoria da probabilidade
 - ▶ Considerando os dados abaixo sobre doenças respiratórias, com os eventos de complicações graves (A) e de infecções assintomáticas ($\neg B$)
 - ▶ Qual é a probabilidade $P[A|B]$ de desenvolver a forma grave da doença estando assintomático?

| | | | |
|----------|----|----------|-----|
| | B | $\neg B$ | |
| A | 2 | 48 | 50 |
| $\neg A$ | 24 | 33 | 57 |
| | 26 | 81 | 107 |

Introdução

- ▶ Teoria da probabilidade
 - ▶ Considerando os dados abaixo sobre doenças respiratórias, com os eventos de complicações graves (A) e de infecções assintomáticas ($\neg B$)
 - ▶ Qual é a probabilidade $P[A|B]$ de desenvolver a forma grave da doença estando assintomático?

| | B | $\neg B$ | |
|----------|-----|----------|-----|
| A | 2 | 48 | 50 |
| $\neg A$ | 24 | 33 | 57 |
| | 26 | 81 | 107 |

$$\begin{aligned}P[A|B] &= \frac{P[B|A] \times P[A]}{P[B]} \\&= \frac{\frac{2}{50} \times \frac{50}{107}}{\frac{26}{107}} \\&= \frac{2}{26} \\&\approx 7,7\%\end{aligned}$$

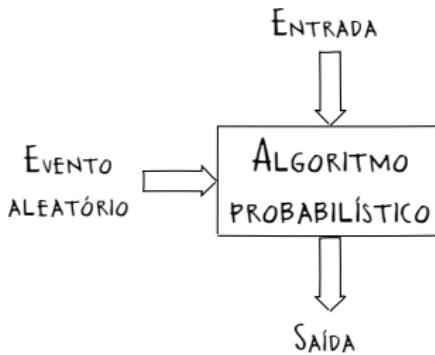
Introdução

- ▶ Algoritmos determinísticos × probabilísticos
 - ▶ Em um algoritmo determinístico, cada passo de sua execução é predeterminado, sempre gerando a mesma saída para uma entrada específica



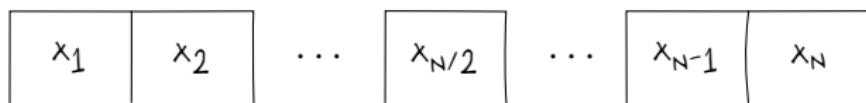
Introdução

- ▶ Algoritmos determinísticos × probabilísticos
 - ▶ Em um algoritmo determinístico, cada passo de sua execução é predeterminado, sempre gerando a mesma saída para uma entrada específica
 - ▶ Já os algoritmos probabilísticos não dependem somente da entrada fornecida, mas também de eventos aleatórios na execução de suas operações



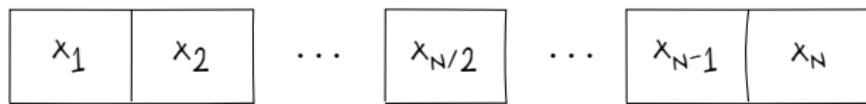
Introdução

- ▶ Algoritmos determinísticos × probabilísticos
 - ▶ Considerando um conjunto de números não ordenados x_1, x_2, \dots, x_n , selecione um número que pertença a metade superior do conjunto, ou seja, um número que é maior ou igual do que a mediana



Introdução

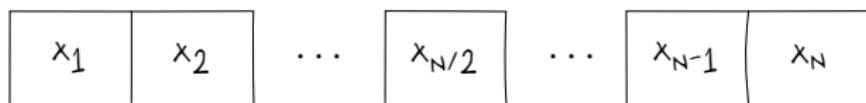
- ▶ Algoritmos determinísticos × probabilísticos
 - ▶ Considerando um conjunto de números não ordenados x_1, x_2, \dots, x_n , selecione um número que pertença a metade superior do conjunto, ou seja, um número que é maior ou igual do que a mediana



DETERMINÍSTICO: ESCOLHE O MAIOR VALOR DE $N/2$
COMPARAÇÕES ENTRE $N/2 + 1$ NÚMEROS DO VETOR

Introdução

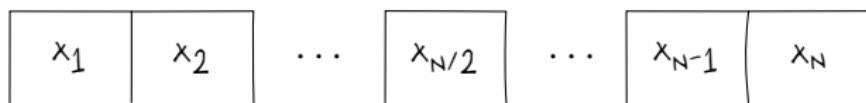
- ▶ Algoritmos determinísticos × probabilísticos
 - ▶ Considerando um conjunto de números não ordenados x_1, x_2, \dots, x_n , selecione um número que pertença a metade superior do conjunto, ou seja, um número que é maior ou igual do que a mediana



PROBABILÍSTICO: SÃO ESCOLHIDOS K NÚMEROS
QUE POSSUEM PELO MENOS 50% DE CHANCE
DE ESTAREM NA METADE SUPERIOR DO VETOR

Introdução

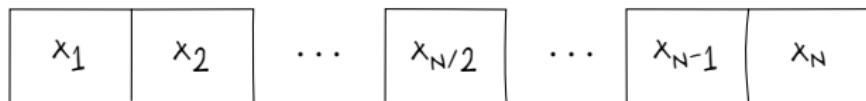
- ▶ Algoritmos determinísticos × probabilísticos
 - ▶ Considerando um conjunto de números não ordenados x_1, x_2, \dots, x_n , selecione um número que pertença a metade superior do conjunto, ou seja, um número que é maior ou igual do que a mediana



A PROBABILIDADE DE NENHUM DESTES K NÚMEROS
ESTAREM NA METADE SUPERIOR DO CONJUNTO É DE $0,5^k$

Introdução

- ▶ Algoritmos determinísticos × probabilísticos
 - ▶ Considerando um conjunto de números não ordenados x_1, x_2, \dots, x_n , selecione um número que pertença a metade superior do conjunto, ou seja, um número que é maior ou igual do que a mediana



COM $K - 1$ COMPARAÇÕES, A PROBABILIDADE DESTE ALGORITMO ESCOLHER CORRETAMENTE O NÚMERO DA METADE SUPERIOR DO CONJUNTO É DE $1 - 0,5^K$

Introdução

- ▶ Abordagens Monte Carlo e Las Vegas
 - ▶ Monte Carlo: pode gerar um resultado incorreto com baixa probabilidade, mas apresenta um tempo de execução que pode ser melhor do que seu respectivo algoritmo determinístico mais eficiente

Introdução

- ▶ Abordagens Monte Carlo e Las Vegas
 - ▶ Monte Carlo: pode gerar um resultado incorreto com baixa probabilidade, mas apresenta um tempo de execução que pode ser melhor do que seu respectivo algoritmo determinístico mais eficiente
 - ▶ Las Vegas: a solução gerada sempre é correta, entretanto, a quantidade de passos não é garantida, devendo ser utilizado quando o tempo de execução esperado é baixo

Introdução

- ▶ Abordagens Monte Carlo e Las Vegas
 - ▶ Monte Carlo: pode gerar um resultado incorreto com baixa probabilidade, mas apresenta um tempo de execução que pode ser melhor do que seu respectivo algoritmo determinístico mais eficiente
 - ▶ Las Vegas: a solução gerada sempre é correta, entretanto, a quantidade de passos não é garantida, devendo ser utilizado quando o tempo de execução esperado é baixo

| | MONTE CARLO | LAS VEGAS |
|---------|----------------|----------------|
| SOLUÇÃO | PROBABILÍSTICA | DETERMINÍSTICA |
| TEMPO | DETERMINÍSTICO | PROBABILÍSTICO |

Algoritmos probabilísticos

- ▶ Método da congruência linear
 - ▶ É essencial para a implementação eficiente dos algoritmos probabilísticos, uma vez que as mesmas sequências de números pseudoaleatórios são criadas deterministicamente a partir de um valor inicial (*seed*)

$$R(i) = [a \times R(i-1) + b] \bmod n$$

Algoritmos probabilísticos

- ▶ Método da congruência linear
 - ▶ É essencial para a implementação eficiente dos algoritmos probabilísticos, uma vez que as mesmas sequências de números pseudoaleatórios são criadas deterministicamente a partir de um valor inicial (*seed*)

$$R(i) = [a \times R(i-1) + b] \bmod n$$

A semente é definida por $R(0)$

Algoritmos probabilísticos

- ▶ Método da congruência linear
 - ▶ É essencial para a implementação eficiente dos algoritmos probabilísticos, uma vez que as mesmas sequências de números pseudoaleatórios são criadas deterministicamente a partir de um valor inicial (*seed*)

$$R(i) = [a \times R(i-1) + b] \bmod n$$

Os parâmetros a e b devem ser escolhidos adequadamente para evitar a repetição dos números gerados na sequência

Algoritmos probabilísticos

- ▶ Método da congruência linear
 - ▶ É essencial para a implementação eficiente dos algoritmos probabilísticos, uma vez que as mesmas sequências de números pseudoaleatórios são criadas deterministicamente a partir de um valor inicial (*seed*)

$$R(i) = [a \times R(i-1) + b] \bmod n$$

Não é adequado para aplicações criptográficas!

Algoritmos probabilísticos

- ▶ Método da congruência linear
 - ▶ Função `rand()` da biblioteca `stdlib` de C/C++: retorna os 15 bits mais significativos $R(i)_{30:16}$
- $$R(i) = [1103515245 \times R(i-1) + 12345] \bmod 2^{31}$$

Algoritmos probabilísticos

- ▶ Método da congruência linear

- ▶ Função `rand()` da biblioteca `stdlib` de C/C++: retorna os 15 bits mais significativos $R(i)_{30:16}$

$$R(i) = [1103515245 \times R(i-1) + 12345] \bmod 2^{31}$$

- ▶ Classe `java.util.Random`: pode gerar um número inteiro de 32 bits de $R(i)_{47:16}$

$$R(i) = [25214903917 \times R(i-1) + 11] \bmod 2^{48}$$

Algoritmos probabilísticos

► Método da congruência linear

```
1 // Variável global do próximo número R(i)
2 static unsigned long next = 1;
3 // Procedimento da semente
4 void srand(unsigned seed) {
5     // R(0) = semente
6     next = seed;
7 }
8 // Função pseudoaleatória
9 int rand() {
10    // Calculando R(i) a partir de R(i - 1)
11    next = (1103515245 * next) + 12345;
12    // Retornando R(i)[30:16]
13    return ((next >> 16) & 0x7FFF);
14 }
```

Algoritmos probabilísticos

► Método da congruência linear

```
1 // Variável global do próximo número R(i)
2 static unsigned long next = 1;
3 // Procedimento da semente
4 void srand(unsigned seed) {
5     // R(0) = semente
6     next = seed;
7 }
8 // Função pseudoaleatória
9 int rand() {
10    // Calculando R(i) a partir de R(i - 1)
11    next = (1103515245 * next) + 12345;
12    // Retornando R(i)[30:16]
13    return ((next >> 16) & 0x7FFF);
14 }
```

$$\begin{array}{ll} \text{rand}() = 6727 \\ \text{srand}(123) \rightarrow \text{rand}() = 22524 \\ \text{rand}() = 25453 \end{array}$$

Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo
 - ▶ Considere a busca de um caractere x em uma cadeia com n elementos, executando no máximo k iterações para encontrar o símbolo procurado



Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo
 - ▶ Considere a busca de um caractere x em uma cadeia com n elementos, executando no máximo k iterações para encontrar o símbolo procurado



$$P[\text{NÃO SER } x] = \frac{2^b - 1}{2^b} = 0,9961$$

Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo
 - ▶ Considere a busca de um caractere x em uma cadeia com n elementos, executando no máximo k iterações para encontrar o símbolo procurado



$$P[\text{NÃO SER } x \text{ POR } k \text{ VEZES}] = P[\text{NÃO SER } x]^k = 0,9961^k$$

Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo
 - ▶ Considere a busca de um caractere x em uma cadeia com n elementos, executando no máximo k iterações para encontrar o símbolo procurado



$$P[\text{NÃO SER } x \text{ POR } k \text{ VEZES}] = 0,9961^k$$

Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo
 - ▶ Considere a busca de um caractere x em uma cadeia com n elementos, executando no máximo k iterações para encontrar o símbolo procurado



$$P[\text{ENCONTRAR } x] = 1 - P[\text{NÃO SER } x \text{ POR } k \text{ VEZES}]$$

Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo
 - ▶ Considere a busca de um caractere x em uma cadeia com n elementos, executando no máximo k iterações para encontrar o símbolo procurado



$$P[\text{ENCONTRAR } x] = 1 - P[\text{NÃO SER } x]^k$$

Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo
 - ▶ Considere a busca de um caractere x em uma cadeia com n elementos, executando no máximo k iterações para encontrar o símbolo procurado



$$P[\text{ENCONTRAR } x] = 1 - 0,9961^k$$

Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo
 - ▶ Considere a busca de um caractere x em uma cadeia com n elementos, executando no máximo k iterações para encontrar o símbolo procurado



$$P[\text{ENCONTRAR } x] = 1 - 0,9961^k$$

$$\stackrel{\uparrow k}{\longrightarrow} \stackrel{\uparrow}{P}[\text{ENCONTRAR } x]$$

Algoritmos probabilísticos

► Abordagem Monte Carlo

- Considere a busca de um caractere x em uma cadeia com n elementos, executando no máximo k iterações para encontrar o símbolo procurado

```
1 // Função busca por caractere (Monte Carlo)
2 int32_t bmc(char* s, uint32_t n, char x, uint32_t k) {
3     // Índice do resultado
4     int32_t r = -1;
5     // k tentativas
6     for(uint32_t i = 0; i < k && r == -1; i++) {
7         // Calculando posição pseudoaleatória
8         uint32_t pos = rand() % n;
9         // Checando se caractere é igual a x
10        if(s[pos] == x) r = pos;
11    }
12    // Retornando resultado
13    return r;
14 }
```

Algoritmos probabilísticos

► Abordagem Monte Carlo

- Considere a busca de um caractere x em uma cadeia com n elementos, executando no máximo k iterações para encontrar o símbolo procurado

```
1 // Função busca por caractere (Monte Carlo)
2 int32_t bmc(char* s, uint32_t n, char x, uint32_t k) {
3     // Índice do resultado
4     int32_t r = -1;
5     // k tentativas
6     for(uint32_t i = 0; i < k && r == -1; i++) {
7         // Calculando posição pseudoaleatória
8         uint32_t pos = rand() % n;
9         // Checando se caractere é igual a x
10        if(s[pos] == x) r = pos;
11    }
12    // Retornando resultado
13    return r;
14 }
```

Espaço $\Theta(n)$ e tempo entre $\Omega(1)$ e $O(k)$

Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo

- ▶ No cenário de busca do caractere x em uma sequência de dados com 500 MB (500×10^6 caracteres), a busca probabilística é ajustada para realizar até 1.000 comparações (k)

$$\begin{aligned} P[\text{encontrar } x] &= 1 - (0,9961)^{10^3} \\ &= 1 - 0,02 \\ &= 0,98 \end{aligned}$$

Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo
 - ▶ No cenário de busca do caractere x em uma sequência de dados com 500 MB (500×10^6 caracteres), a busca probabilística é ajustada para realizar até 1.000 comparações (k)

$$\begin{aligned}P[\text{encontrar } x] &= 1 - (0,9961)^{10^3} \\&= 1 - 0,02 \\&= 0,98\end{aligned}$$

Com até 1.000 tentativas, existe uma probabilidade de sucesso de 98% para encontrar x

Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo
 - ▶ Pequeno teorema de Fermat

$$\text{primo}(p) \wedge 1 \leq a < p \rightarrow a^{p-1} \equiv 1 \pmod{p}$$

Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo
 - ▶ Pequeno teorema de Fermat

$$\text{primo}(p) \wedge 1 \leq a < p \rightarrow a^{p-1} \equiv 1 \pmod{p}$$

- ▶ Teste de primalidade de Fermat

$$\text{primo_fermat}(n) = \begin{cases} a^{n-1} \equiv 1 \pmod{n} & , \text{ provável primo} \\ a^{n-1} \not\equiv 1 \pmod{n} & , \text{ composto} \end{cases}$$

Algoritmos probabilísticos

- ▶ Abordagem Monte Carlo
 - ▶ Pequeno teorema de Fermat

$$\text{primo}(p) \wedge 1 \leq a < p \rightarrow a^{p-1} \equiv 1 \pmod{p}$$

- ▶ Teste de primalidade de Fermat

$$\text{primo_fermat}(n) = \begin{cases} a^{n-1} \equiv 1 \pmod{n} & , \text{ provável primo} \\ a^{n-1} \not\equiv 1 \pmod{n} & , \text{ composto} \end{cases}$$

Para $n < 10.000$ existem apenas 22 valores que podem gerar um resultado incorreto (0,22%)

Algoritmos probabilísticos

► Abordagem Monte Carlo

- O teste de primalidade de Fermat executa por k iterações para determinar se n é um provável primo

```
1 // Teste de primalidade de Fermat
2 uint8_t primo_fermat(uint64_t n, uint32_t k) {
3     // 1 = provável primo
4     uint8_t r = 1;
5     // k tentativas
6     for(uint32_t i = 0; i < k && r == 1; i++) {
7         // Obtendo a (1 <= a < n)
8         uint64_t a = 1 + rand64() % (n - 1);
9         // r = a ^ (p - 1)
10        r = expmod(a, n - 1, n);
11    }
12    // Retornando resultado
13    return r;
14 }
```

Algoritmos probabilísticos

► Abordagem Monte Carlo

- O teste de primalidade de Fermat executa por k iterações para determinar se n é um provável primo

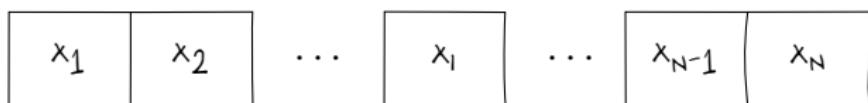
```
1 // Teste de primalidade de Fermat
2 uint8_t primo_fermat(uint64_t n, uint32_t k) {
3     // 1 = provável primo
4     uint8_t r = 1;
5     // k tentativas
6     for(uint32_t i = 0; i < k && r == 1; i++) {
7         // Obtendo a (1 <= a < n)
8         uint64_t a = 1 + rand64() % (n - 1);
9         // r = a ^ (p - 1)
10        r = expmod(a, n - 1, n);
11    }
12    // Retornando resultado
13    return r;
14 }
```

Espaço $\Theta(1)$ e tempo entre $\Omega(\log_2 n)$ e $O(k \log_2 n)$

Algoritmos probabilísticos

► Abordagem Las Vegas

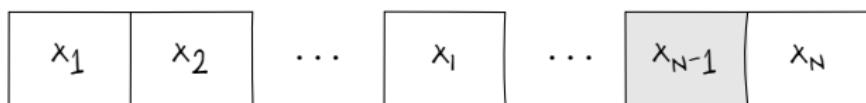
- Considere a busca de um caractere x em uma cadeia com n elementos, executando comparações em posições pseudoaleatórias até encontrar x



Algoritmos probabilísticos

► Abordagem Las Vegas

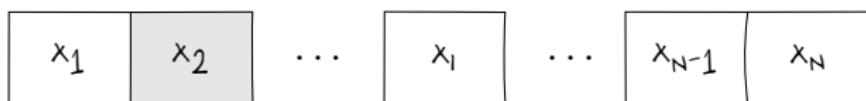
- Considere a busca de um caractere x em uma cadeia com n elementos, executando comparações em posições pseudoaleatórias até encontrar x



Algoritmos probabilísticos

► Abordagem Las Vegas

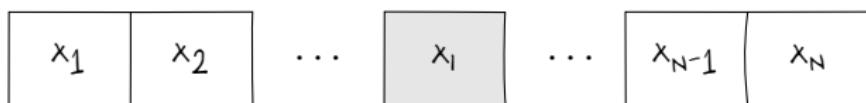
- Considere a busca de um caractere x em uma cadeia com n elementos, executando comparações em posições pseudoaleatórias até encontrar x



Algoritmos probabilísticos

► Abordagem Las Vegas

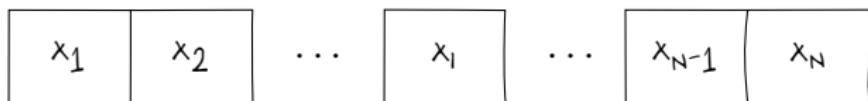
- Considere a busca de um caractere x em uma cadeia com n elementos, executando comparações em posições pseudoaleatórias até encontrar x



Algoritmos probabilísticos

► Abordagem Las Vegas

- Considere a busca de um caractere x em uma cadeia com n elementos, executando comparações em posições pseudoaleatórias até encontrar x



Se estiver na cadeia, o símbolo sempre é encontrado

Algoritmos probabilísticos

► Abordagem Las Vegas

- Considere a busca de um caractere x em uma cadeia com n elementos, executando comparações em posições pseudoaleatórias até encontrar x

```
1 // Função busca por caractere (Las Vegas)
2 int32_t blv(char* s, uint32_t n, char x) {
3     // Índice do resultado
4     int32_t r = -1;
5     // Enquanto x não foi encontrado
6     while(r == -1 && pendentes(n)) {
7         // Calculando posição pseudoaleatória
8         uint32_t pos = rand_pendentes();
9         // Comparando e marcando posição
10        if(comparar(s, pos, x)) r = pos;
11    }
12    // Retornando resultado
13    return r;
14 }
```

Algoritmos probabilísticos

► Abordagem Las Vegas

- Considere a busca de um caractere x em uma cadeia com n elementos, executando comparações em posições pseudoaleatórias até encontrar x

```
1 // Função busca por caractere (Las Vegas)
2 int32_t blv(char* s, uint32_t n, char x) {
3     // Índice do resultado
4     int32_t r = -1;
5     // Enquanto x não foi encontrado
6     while(r == -1 && pendentes(n)) {
7         // Calculando posição pseudoaleatória
8         uint32_t pos = rand_pendentes();
9         // Comparando e marcando posição
10        if(comparar(s, pos, x)) r = pos;
11    }
12    // Retornando resultado
13    return r;
14 }
```

Espaço $\Theta(n)$ e tempo entre $\Omega(1)$ e $O(n)$

Algoritmos probabilísticos

- ▶ Abordagem Las Vegas
 - ▶ Particionamento do Quicksort

```
1 // Particionamento randômico do Hoare
2 void hoare_lv(int32_t* V, int32_t i, int32_t j) {
3     // Troca do pivô por aleatório
4     trocar(&V[i], &V[i + (rand() % (j - i + 1))]);
5     // Chamada do particionamento
6     return hoare(V, i, j);
7 }
8 // Particionamento randômico do Lomuto
9 void lomuto_lv(int32_t* V, int32_t i, int32_t j) {
10    // Troca do pivô por aleatório
11    trocar(&V[j], &V[i + (rand() % (j - i + 1))]);
12    // Chamada do particionamento
13    return lomuto(V, i, j);
14 }
```

Algoritmos probabilísticos

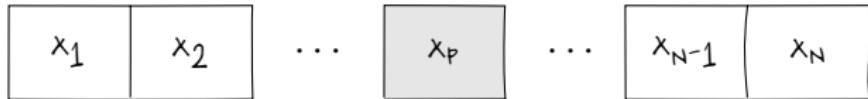
- ▶ Abordagem Las Vegas
 - ▶ Particionamento do Quicksort

```
1 // Particionamento randômico do Hoare
2 void hoare_lv(int32_t* V, int32_t i, int32_t j) {
3     // Troca do pivô por aleatório
4     trocar(&V[i], &V[i + (rand() % (j - i + 1))]);
5     // Chamada do particionamento
6     return hoare(V, i, j);
7 }
8 // Particionamento randômico do Lomuto
9 void lomuto_lv(int32_t* V, int32_t i, int32_t j) {
10    // Troca do pivô por aleatório
11    trocar(&V[j], &V[i + (rand() % (j - i + 1))]);
12    // Chamada do particionamento
13    return lomuto(V, i, j);
14 }
```

A escolha pseudoaleatória do pivô busca
reduzir as chances de escolha do maior ou menor
elemento da partição (pior caso)

Algoritmos probabilísticos

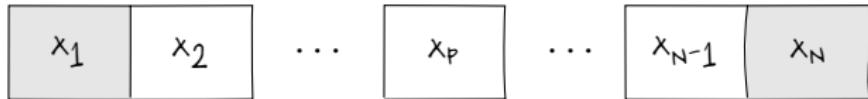
- ▶ Abordagem Las Vegas
 - ▶ Particionamento do Quicksort



$$P[x_p = \text{MEDIANA}] = \frac{1}{N}$$

Algoritmos probabilísticos

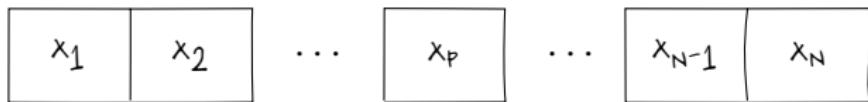
- ▶ Abordagem Las Vegas
 - ▶ Particionamento do Quicksort



$$P[x_1 = 1] = P[x_1 = N] = \frac{1}{N}$$

Algoritmos probabilísticos

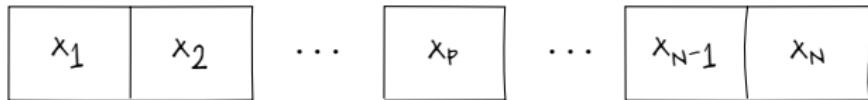
- ▶ Abordagem Las Vegas
 - ▶ Particionamento do Quicksort



$$P[x_p = \text{MEDIANA} \wedge x_p = i] = \frac{1}{n} \times \frac{1}{n}$$

Algoritmos probabilísticos

- ▶ Abordagem Las Vegas
 - ▶ Particionamento do Quicksort



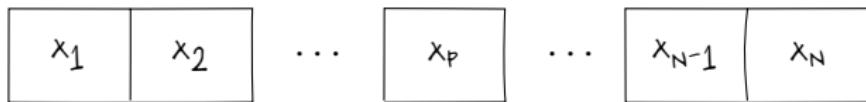
$$P[x_p = \text{MEDIANA}] = \frac{1}{n}$$

>

$$P[x_p = \text{MEDIANA} \wedge x_p = i] = \frac{1}{n} \times \frac{1}{n}$$

Algoritmos probabilísticos

- ▶ Abordagem Las Vegas
 - ▶ Particionamento do Quicksort



QUALQUER QUE SEJA O PIVÔ ESCOLHIDO,
A ORDENAÇÃO SERÁ REALIZADA CORRETAMENTE,
COM UM TEMPO DE EXECUÇÃO PSEUDOALEATÓRIO