ALGORITMOS RANDOMIZADOS

Marcus Ritt

Outline

- Ideia: Um algoritmo randomizado usa *eventos aleatórios* na sua execução.
- Modelos computacionais:
  - Máquina de Turing probabilística
  - Máquina RAM com um comando `random(S)` que retorne um elemento aleatório do conjunto S.

uniforme

random(10,13)

random(11, 43, 45, 5, 63)

Porque aceitar probabilidades?

- Probabilidade morrer caindo da cama: $1/2 \times 10^6$ (Roach & Pieper 2007).
- Probabilidade acertar 6 números de 60 na mega-sena: $1/50063860$.
- Probabilidade que a memória falha: em memória moderna temos 1000 FIT/MBit, i.e.\ 6×10^{-7} erros por segundo num memória de 256 MB.
- Probabilidade que um meteorito destrói um computador em cada milissegundo: $\geq 2^{-100}$ (supondo que cada milênio ao menos um meteorito destrói uma área de $100\,m^2$).

$\Longrightarrow$ Um algoritmo que retorna uma resposta falsa com baixa probabilidade é aceitável

Possíveis vantagens: um algoritmo

- mais simples;
- mais eficiente: para alguns problemas, um algoritmo randomizado é o mais eficiente conhecido;
- mais robusto: algoritmos randomizados podem ser menos dependente da distribuição das entradas.
- a única alternativa: para alguns problemas, conhecemos só algoritmos randomizados.

Revisão: Probabilidade

- *Espaço amostral* finito Ω de *eventos elementares* $e \in \Omega$.
- Distribuição de probabilidade $\Pr(e)$ tal que

$$\Pr(e) \geq 0; \qquad \sum_{e \in \Omega} \Pr(e) = 1$$

- *Eventos* (compostos) $E \subseteq \Omega$ com probabilidade

$$\Pr(E) = \sum_{e \in E} \Pr(e)$$

Para um dado sem bias temos $\Omega = \{1, 2, 3, 4, 5, 6\}$ e $\Pr(i) = 1/6$.
O evento $\text{Par} = \{2, 4, 6\}$ tem probabilidade
$\Pr(\text{Par}) = \sum_{e \in \text{Par}} \Pr(e) = 1/2$.

Revisão: Probabilidade

- *Variável aleatória*

$$X : \Omega \to \mathbb{N}$$

$\mathbb{N} = \{0, 1, 2, \ldots\}$

- Escrevemos $\Pr(X = i)$ para $\Pr(X^{-1}(i))$.
- Variáveis aleatórias *independentes*

$$P[X = x \text{ e } Y = y] = P[X = x] \cdot P[Y = y]$$

- *Valor esperado*

$$E[X] = \sum_{e \in \Omega} \Pr(e) X(e) = \sum_{i \geq 0} i \Pr(X = i)$$

- Linearidade do valor esperado: Para variáveis aleatórias X, Y

$$E[X + Y] = E[X] + E[Y]$$

Seja X a variável aleatória que denota o número sorteado, e Y a variável aleatória tal que

$Y = [\text{a face em cima do dado tem um ponto no meio}]$

$$E[X] = \sum_{i \geq 0} \Pr(X = i) i = 1/6 \sum_{1 \leq i \leq 6} i = 21/6 = 7/2$$

$$E[Y] = \sum_{i \geq 0} \Pr(Y = i) i = \Pr(Y = 1) = 1/2$$

$$E[X + Y] = E[X] + E[Y] = 4$$

$\Sigma = \{0,1\}$ $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots\}$

Definição 2.1

Seja Σ algum alfabeto e $R(\alpha, \beta)$ a classe de linguagens $L \subseteq \Sigma^*$ tal que existe um algoritmo de decisão em tempo polinomial A que satisfaz

- $x \in L \Rightarrow \Pr(A(x) = \text{sim}) \geq \alpha.$
- $x \notin L \Rightarrow \Pr(A(x) = \text{não}) \geq \beta.$

P

(A probabilidade é sobre todas sequências de bits aleatórios r. Como o algoritmo executa em tempo polinomial no tamanho da entrada $|x|$, o número de bits aleatórios $|r|$ é polinomial em $|x|$ também.)

R(1,1) nos dá algo correto / com a randomização

Definição 2.2

(Classes de complexidade)

- $RP := R(1/2, 1)$ (randomized polynomial), dos problemas que possuem um algoritmo com erro unilateral (no lado do "sim"); a classe $co - RP = R(1, 1/2)$ consiste dos problemas com erro no lado de "não";
- $ZPP := RP \cap co - RP$ (zero-error probabilistic polynomial) dos problemas que possuem algoritmo randomizado sem erro;
- $PP := \bigcup_{\epsilon \in (0,1/2]} R(1/2 + \epsilon, 1/2 + \epsilon)$ (probabilistic polynomial), dos problemas com erro $1/2 + \epsilon$ nos dois lados; e
- $BPP := R(2/3, 2/3)$ (bounded-error probabilistic polynomial), dos problemas com erro $1/3$ nos dois lados.

Monte Carlo algoritmos que respondem corretamente somente com uma certa probabilidade

Las Vegas algoritmos que usam randomização somente internamente, mas respondem sempre corretamente

Exemplo

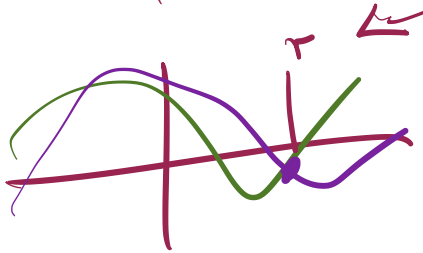
- Dado dois polinômios $p(x)$ e $q(x)$ de grau máximo d, como saber se $p(x) \equiv q(x)$?
- É simples responder por comparação de coeficientes em tempo $O(n)$
  - os dois na forma canônica $p(x) = \sum_{0 \leq i \leq d} p_i x^i$ ou
  - na forma fatorada $p(x) = \prod_{1 \leq i \leq d} (x - r_i)$ isso
- E caso contrário?
  - Converter para a forma canônica pode custar $\Theta(d^2)$ multiplicações.

Exemplo

identico(p,q) :=
  Seleciona número aleatório r *em* $[1, 100d]$.
  Caso $p(r) = q(r)$ *retorne "sim".*
  Caso $p(r) \neq q(r)$ *retorne "não".*

$x \in L$ | (p,q) pos. $\Rightarrow$ goren. "sim"
$x \notin L$ | (p,q) neg. $\Rightarrow$ erro

R(1, p)



Exemplo

- Caso $p(x) \equiv q(x)$, o algoritmo responde "sim" com certeza.
- Caso contrário a resposta pode ser errada, se $p(r) = q(r)$ por acaso. Qual a probabilidade disso?
- $p(x) - q(x)$ é um polinômio de grau d e possui no máximo d raízes.
  - Portanto, a probabilidade de encontrar um r tal que $p(r) = q(r)$, caso $p \not\equiv q$ é $d/100d = 1/100$.
  - Isso demonstra que o teste de identidade pertence à classe $co - RP$.

Observação 2.1

É uma pergunta em aberta se o teste de identidade pertence a P. $\Diamond$

- Dissatisfeito com a probabilidade de $1/100$ no exemplo?
- Repete o algoritmo k vezes, e responde "sim" somente se todas k repetições responderam "sim".
- A probabilidade erradamente responder "não" para polinômios idênticos agora é $(1/100)^k$, i.e. ela diminui exponencialmente com o número de repetições.

- Essa técnica é uma *amplificação* da probabilidade de obter a solução correta.
- Ela pode ser aplicada para melhorar a qualidade de algoritmos em todas classes "Monte Carlo".
- Com um número constante de repetições, obtemos uma probabilidade baixa nas classes RP, co-RP e BPP.
- Isso não se aplica a PP: é possível que ϵ diminui exponencialmente com o tamanho da instância.

$PP = \bigcup R(1/2+\epsilon, 1/2+\epsilon)$

$\bigcup R(2^{-3}+\epsilon, 2^{-3}+\epsilon)$

Teorema 3.1

$R(\alpha, 1) = R(\beta, 1)$ para $0 < \alpha, \beta < 1$.

$\alpha < \beta$. $R(\beta,1) \subseteq R(\alpha,1)$

a.l: $R(\alpha,1) \subseteq R(\beta,1)$:

$L \in R(\alpha,1) \Rightarrow L \in R(\beta,1)$

$\exists$ alg. A, pol., rand. $x \in L$: sim $Pr \geq \alpha$
não $Pr = 1$

a.l. $\exists$ alg. A', pol., rand: $x \in L$: sim $Pr \geq \beta$
não $Pr = 1$.

Amplificação: repete k vezes
resposta sim: (se todos ex. "dão sim")

A' ↗ Responde não : C.C.

$Pr(A'(x) = \text{não}) = 1$ Caso $x \notin L$

Caso: $x \in L$ $\quad Pr[A(x) = \text{"sim"}] \geq \alpha$ ↙

$Pr[A'(x) = \text{"sim"}] \geq \boxed{1 - (1-\alpha)^k}$

$1-(1-\alpha)^k$ $\quad \log_{1-\alpha} 1-\beta$

$k \geq \ln(1-\beta) / \ln(1-\alpha)$

$1-(1-\alpha)^k \geq \beta$ $\quad 1-(1-\alpha)^{\ln \frac{(1-\beta)}{(1-\alpha)}} =$

$1-(1-\alpha)^{\log_{1-\alpha} 1-\beta} = 1-(1-\beta) = \beta$

$RP = R(\frac{1}{2}, 1)$

Corolário 3.1

$RP = R(\alpha, 1)$ para $0 < \alpha < 1$.

$co\text{-}RP = R(1, \alpha) \quad \alpha \in (0, 1)$.

$R(1, \alpha) = R(1, \beta)$, $\alpha, \beta \in (0, 1)$

Teorema 3.2

$R(\alpha, \alpha) = R(\beta, \beta)$ para $1/2 < \alpha, \beta$.

$\alpha < \beta$ $\quad R(\beta,\beta) \subseteq R(\alpha,\alpha)$.

a.c.: $R(\alpha,\alpha) \subseteq R(\beta,\beta)$

$\downarrow$

$d \in R(\alpha,\alpha) \Rightarrow d \in R(\beta,\beta)$

$\Rightarrow \nexists$ alg A $\quad x \in d \ \ P_r \geq \alpha$
$x \notin d \ \ P_r \geq \alpha$.

Alg. A': repete alg. A k vezes
resp. "sim" caso a maioria rep. "sim"
rep. outra caso contr.

$\Pr(A'(x) = \text{"sim"}) = \Pr(A(x) = \text{"sim" mais que } \lfloor k/2 \rfloor + 1 \text{ vezes})$

$\alpha > 1/2$

$\geq 1 - e^{-2k(\alpha - 1/2)^2}$ [Chernoff bounds]

$\left(e^{-2(\alpha-1/2)^2}\right)^k$

$k \geq \ln(b-1)/2(\alpha-1/2)^2 \geq \beta.$

"mais": idêntico.

Corolário 3.2
$BPP = R(\alpha, \alpha)$ para $1/2 < \alpha$.

$BPP = R(2/3, 2/3)$
$=$
$R(3/4, 3/4)$

$BPP = R(\alpha, b)$? $\alpha, b > 1/2$

Observação 3.1

Os resultados acima são válidos ainda caso o erro dimiui polinomialmente com o tamanho da instância, i.e.~$\alpha, \beta \geq n^{-c}$ no caso do teorema 3.1 e $\alpha, \beta \geq 1/2 + n^{-c}$ no caso do teorema 3.2 para um constante c (ver por exemplo Arora \ Barak (2009)). $\Diamond$

Definição 4.1

Um algoritmo A é *honesto* se

- ele responde ou "sim", ou "não" ou "não sei", - $\Pr(A(x) = \text{não sei}) \leq 1/2$, e - no caso ele responde, ele não erra, i.e., para x tal que $A(x) \neq$ "não sei" temos ~~$A(x) =$ "sim" $\iff x \in L$~~.

Uma linguagem é honesta caso ela possui um algoritmo honesto.
Com isso também podemos falar da classe das linguagens honestas.

$A(x) = \text{"sim"} \Rightarrow x \in L$
$A(x) = \text{"não"} \Rightarrow x \notin L$

$ZPP = RP \cap co\text{-}RP$
$= HONESTO$

Teorema 4.1
ZPP é a classe das linguagens honestas.

Lema 4.1
Caso $L \in ZPP$ existe um algoritmo um algoritmo honesto para L.

Lema 4.2
Caso L possui um algoritmo honesto $L \in RP$ e $L \in co - RP$.

4.1 $L \in ZPP = RP \cap co\text{-}RP$

$L \in RP$: $\exists A_1$ sim: $Pr \geq 1/2$; não: $Pr = 1$

$L \in co\text{-}RP$: $\exists A_2$ sim: $Pr = 1$; não: $Pr \geq 1/2$

Alg. $A'(x)$

1) $A_1(x) =$ "não" $\wedge$ $A_2(x) =$ "não" : "não"

a) "não" $\wedge$ "sim" : "não sei"

3) "sim" $\wedge$ "não" : X

4) "sim" $\wedge$ "sim" : "sim"

$Pr(\text{caso } 2)$: $x \in L$: $1/2 \cdot 1$; $x \notin L$: $1 \cdot 1/2$ $\leq 1/2$

4.2

$L \in$ HONESTO

$\Rightarrow L \in RP \Rightarrow R(1/2, 1)$

$\Rightarrow L \in co\text{-}RP$

Seja A um alg. honesto p/ L

$A(x) =$ "sim" : "sim"

$A(x) =$ "não" : "não"

$A(x) =$ "não sei" : "não"

$Pr(\cdot) \geq 1/2$

Definição 4.2

Um algoritmo A é *sem falha* se ele sempre responde "sim" ou "não" corretamente em *tempo polinomial esperado*. Com isso podemos também falar de linguagens sem falha e a classe das linguagens sem falha.

Teorema 4.2

ZPP é a classe das linguagens sem falha.

Lema 4.3

Caso $L \in ZPP$ existe um algoritmo sem falha para L.

Lema 4.4

Caso L possui um algoritmo A sem falha, $L \in RP$ e $L \in co-RP$.

4.3 $L \in ZPP \Rightarrow L \in$ HONESTO

$\exists$ Alg. A honesto;

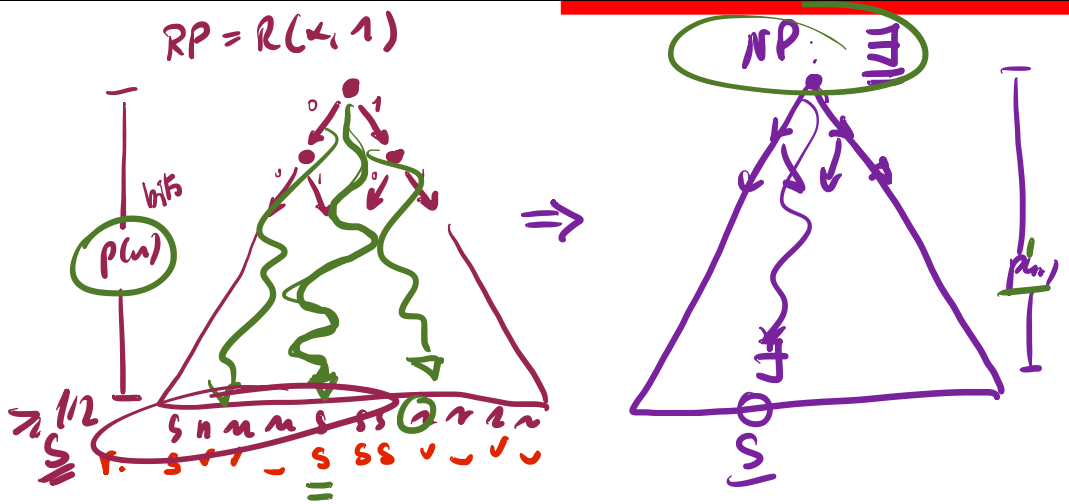
Alg. A': roda A até obter "sim"/"não"

$\sum_{k>0} k \cdot p(n) \cdot 2^{-k} \leq 2p(n)$.

4.4 A um alg. sem falha L tempo esp. $p(n)$

Alg. A': roda A por tempo $2p(n)$ (caso tem resposta: responde igual. C.C.: "não sei"

$Pr(T \geq 2p(n)) \leq_{\text{Markov}} p(n)/2p(n) \leq 1/2$.



Teorema 4.3

$RP \subseteq$ e $co - RP \subseteq co - NP$

$RP \subseteq NP$

$L \in RP \Rightarrow \exists A$

$L \in NP$? gerar todas seq. de $p(n)$ bits aleatórios e depois executar determinísticamente

$co\text{-}RP \subseteq co\text{-}NP$

$co\text{-}NP$: $\forall$

$BPP = co-BPP$

Teorema 4.4

$RP \subseteq BPP$ e $co-RP \subseteq BPP$.

$L \in RP$: $\exists A$ alg. A: $x \in L$: $Pr \geq 1/2$, $x \notin L$: $Pr = 1$. $\Rightarrow$ a.a. $\exists A'$ alg. A' $x \in L$: $Pr \geq 2/3$, $x \notin L$: $Pr > 1/3$

Alg. $A'(x)$: if $A(x) =$ "não" $\wedge$ $A(x) =$ "não": não
c.c.: "sim".

$x \notin L$: $Pr(A'(x) =$ "não"$) = Pr(A(x) =$ "não"$) \times Pr(A(x) =$ "não"$) = 1 \cdot 1 = 1$

$x \in L$: $Pr(A'(x) =$ "sim"$) = (1 - Pr(A'(x) =$ "não"$)$
$= 1 - 1/2 \cdot 1/2 > 2/3$.

- Classe NP: problemas que permitem uma verificação de uma solução em tempo polinomial
- Não-determinísticamente: podemos "chutar" uma solução e verificá-la.
- Se o número de soluções positivas de cada instância é mais que a metade do número total de soluções
  - $\Longrightarrow$ o problema pertence a RP $= R(1/2, 1) = R(\alpha, 1)$
  - podemos gerar uma solução aleatória e testar se ela possui a característica desejada.
- Um problema desse tipo possui uma *abundância de testemunhas*.
  - Isso demonstra a importância de algoritmos randomizados.
  - O teste de equivalência de polinômios acima é um exemplo de abundância de testemunhas.

Relação entre as classes