Algoritmos Aleatorizados

André Vignatti

DINF- UFPR

Introdução

Quem nunca usou números aleatórios dentro de um programa?

2 Motivos Principais:

1. O mundo comporta-se aleatoriamente: algoritmos tradicionais recebem entradas aleatórias.
   - **Análise de caso médio**: ao invés do pior caso, fazemos uma "média" com todas possíveis entradas
2. O algoritmo comporta-se aleatoriamente: o mundo provê (como sempre) a entrada de **pior caso**, mas o algoritmo faz decisões aleatórias
   - Neste caso, a aleatorização é interna do algoritmo. É essa noção de algoritmo aleatorizado que iremos estudar.

O que é aleatorizar?

Aleatorizar

É permitir um algoritmo jogar uma moeda (ou um dado de n faces) em tempo unitário (constante).

- Na prática, usa-se um gerador de números pseudo-aleatórios.

Porque aleatorizar?

São mais **poderosos**. Os algoritmos determinísticos eficientes que produzem resposta correta são casos especiais de:

- algoritmos aleatorizados que produzem com alta probabilidade a resposta correta,
  - Determinísticos: probabilidade $p = 1$ de estar correto.
  - Aleatorizados: probabilidade $0 < p$ *le*1 de estar correto.
- algoritmos aleatorizados que sempre produzem a resposta correta, mas executam eficientemente no tempo esperado.
  - Determinísticos: tempo $T(n)$
  - Aleatorizados: tempo esperado $T(n)$, mas pode haver desvio do esperado).

Porque aleatorizar?

Alguns problemas não resolvidos eficientemente por algoritmos determinísticos podem ser resolvidos por algoritmos probabilísticos.

Muitas vezes...

... produzem soluções mais simples, rápidas ou o único algoritmo conhecido para resolver um certo problema.

Exemplos de aplicações

Quebra de simetria, algoritmos em grafos, quicksort, hashing, balanceamento de carga, criptografia, etc...

Porque aleatorizar?

Uma preocupação...

- Algoritmos aleatorizados $=$ saber MUITA probabilidade.
- De fato, alguns algoritmos requerem ideias complexas de probabilidade.
- Mas nosso objetivo aqui é mostrar que só **com um pouco de probabilidade** dá para entender muitos algoritmos aleatorizados conhecidos.

Problema: Desfazendo Contenção

Desfazendo Contenção

Dados n processos $P_1, \ldots, P_n$, competindo para acessar um BD compartilhado. O tempo é dividido em **rounds**. Se dois ou mais processos acessam o banco de dados no mesmo round, todos processos ficam travados até o final daquele round. Projete um protocolo que garanta que todos processos sejam executados.

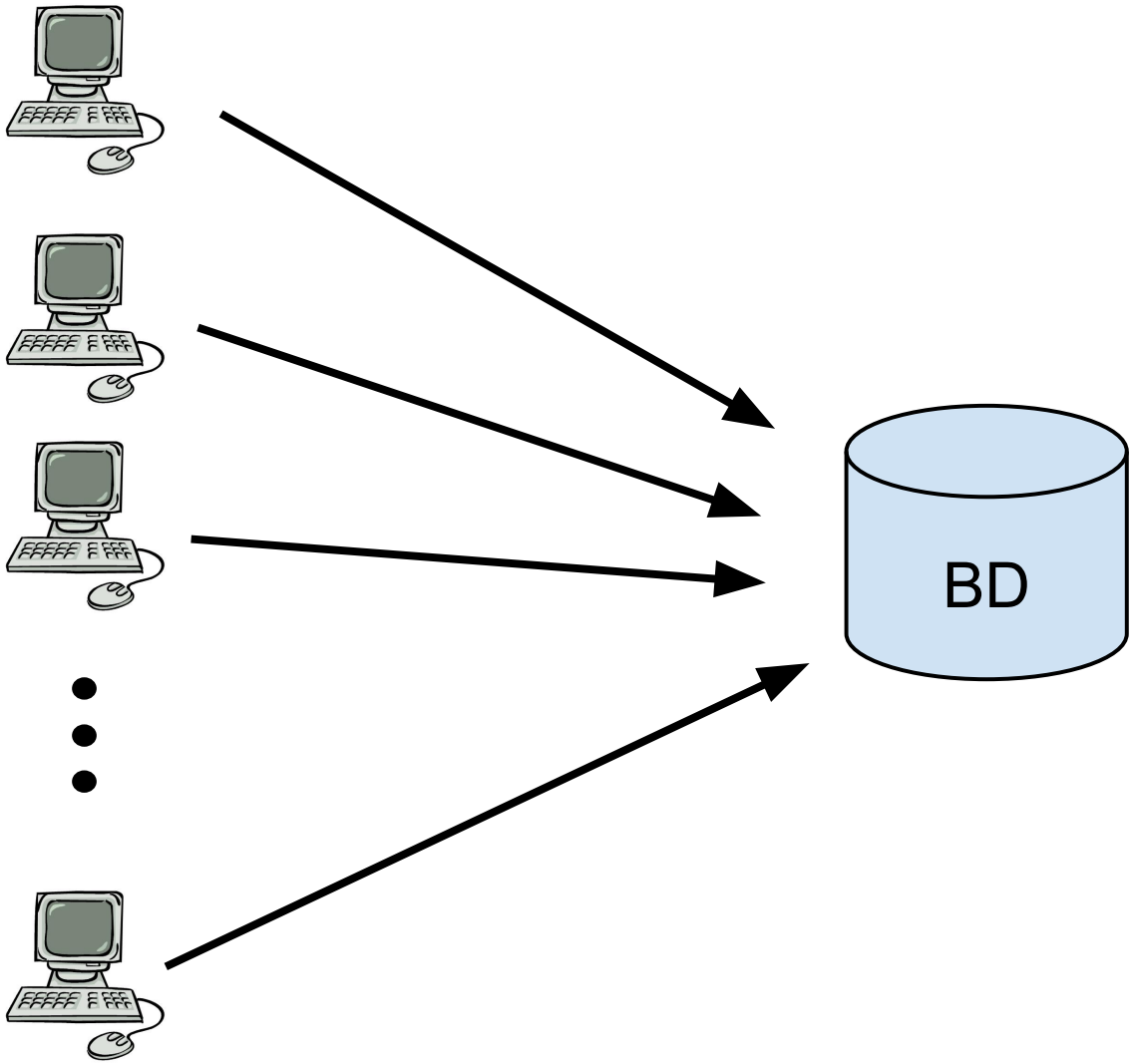
Restrição

Os processos não podem se comunicar.

Desafio

Precisamos de um paradigma de **quebra de simetria**.

Problema: Desfazendo Contenção



Solução

O protocolo

Cada processo tenta acessar o banco de dados num certo round com probabilidade p (iremos definir p depois).

Perguntas:

- Qual a probabilidade de um processo ter sucesso **num único round**?
- Quantos rounds são necessários para **um único processo ter sucesso**?
- Quantos rounds são necessários para **TODOS os processos terem sucesso**?

Eventos

Recomenda-se SEMPRE em algoritmos aleatorizados definir **eventos básicos**. Alguns eventos básicos:

$A[i,t]$: é o **evento** de P_i tentar acessar o banco de dados no round t.

- $Pr\left[A[i,t]\right] = p$.

$\overline{A[i,t]}$ é o **evento** complementar de $A[i,t]$.

- $Pr\left[\overline{A[i,t]}\right] = 1 - Pr\left[A[i,t]\right] = 1 - p$.

Eventos

$S[i,t]$: é o **evento** de P_i ter sucesso em acessar o banco de dados no round t.

- $S[i,t]$ acontece quando, no round t, P_i tenta acessar o banco de dados, e os outros processos não tentam acessar.
- $S[i,t] = A[i,t] \cap \left(\bigcap_{j \neq i} \overline{A[j,t]} \right)$.

Eventos

Todos os eventos na intersecção são independentes, pela definição do protocolo. Assim, podemos simplesmente multiplicar as probabilidades:

- $Pr\left[S[i,t]\right] = Pr\left[A[i,t]\right] \cdot \prod_{j \neq i} Pr\left[\overline{A[j,t]}\right] = p(1-p)^{n-1}$.

- Agora temos uma formulinha bonita para a probabilidade de sucesso.
- Basta "setar" o p com o valor que maximize essa fórmula.
- **PERGUNTA**: Como achar o máximo de $f(p) = p(1-p)^{n-1}$?
- **RESPOSTA**: Faz a derivada igual a zero!
- $f'(p) = (1-p)^{n-1} - (n-1)p(1-p)^{n-2}$ tem valor zero somente quando $p = 1/n$.

Eventos

Assim, fazendo $p = 1/n$ temos $Pr\left[S[i,t]\right] = \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1}$.

Vale a pena ver o que acontece assintoticamente (quando $n \to \infty$):

À medida que n cresce de 2 até ∞:

- $\left(1 - \frac{1}{n}\right)^{n}$ converge crescendo monotonicamente de $\frac{1}{4}$ até $\frac{1}{e}$.
- $\left(1 - \frac{1}{n}\right)^{n-1}$ converge decrescendo monotonicamente de $\frac{1}{2}$ até $\frac{1}{e}$.

Assim, sabemos que $\frac{1}{en} \leq Pr\left[S[i,t]\right] \leq \frac{1}{2n}$. Ou seja, $Pr\left[S[i,t]\right] = \Theta(1/n)$.

Quantos rounds até um processo ter sucesso?

- Vimos que a probabilidade de um processo ter sucesso num único round não é muito grande (principalmente se n é grande).
- E se considerarmos vários rounds?

$F[i,t]$: é o **evento** de P_i falhar em acessar o bando de dados em todos os rounds 1 até t.

- $Pr\left[F[i,t]\right] = Pr\left[\bigcap_{r=1}^{t} \overline{S[i,r]}\right] = \prod_{r=1}^{t} Pr\left[\overline{S[i,r]}\right] = \left(1 - \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1}\right)^{t}.$

Quantos rounds até um processo ter sucesso?

A expressão da probabilidade está correta, mas começa a ficar complicada. Talvez seja melhor pensarmos assintoticamente:

- $Pr\left[F[i,t]\right] = \prod_{r=1}^{t} Pr\left[\overline{S[i,r]}\right] = \prod_{r=1}^{t}\left(1 - Pr\left[S[i,r]\right]\right) \leq \prod_{r=1}^{t}\left(1 - \frac{1}{en}\right) = \left(1 - \frac{1}{en}\right)^{t}$.

Se fizermos $t = en$, podemos substituir diretamente nas equivalências assintóticas descritas anteriormente.

Observação

Como en não é inteiro, vamos fazer $t = \lceil en \rceil$.

Quantos rounds até um processo ter sucesso?

- $Pr\left[F[i,t]\right] \leq \left(1 - \frac{1}{en}\right)^{\lceil en \rceil} \leq \left(1 - \frac{1}{en}\right)^{en} \leq \frac{1}{e}$.

Resumindo...

- Após $t = \lceil en \rceil$ rounds, a probabilidade de um processo não ter sucesso é no máximo e^{-1}.
- Se aumentarmos um pouco o número de rounds, digamos $t = \lceil en \rceil \cdot (c \ln n)$, a probabilidade de falha diminui muito!
- $Pr\left[F[i,t]\right] \leq \left(\left(1 - \frac{1}{en}\right)^{en}\right)^{c \ln n} \leq e^{-c \ln n} = n^{-c}$.

Quantos rounds até um processo ter sucesso?

Então a pergunta: **"Quantos rounds até um processo ter sucesso?"** está respondida! (Yesssssssssssss)

Assintóticamente...

...podemos pensar assim:

- Após $\Theta(n)$ rounds, a probabilidade de P_i não ter sucesso é limitada por uma constante.
- Após $\Theta(n \log n)$ rounds, a probabilidade de P_i não ter sucesso é limitada pelo inverso de uma função polinomial em n.

Quando todos os processos têm sucesso?

Finalmente, perguntamos: Após quantos rounds temos alta probabilidade de que todos os processos acessem o banco de dados ao menos uma vez?

Observação

Como convenção na **grande maioria** da literatura, o termo **alta probabilidade** significa $1 - \frac{1}{n^c}$, para $c \geq 1$.

Quando todos os processos têm sucesso?

Definição: Falha do Protocolo

Dizemos que o protocolo falha após t rounds se algum processo ainda não teve sucesso ao acessar o banco de dados.

F_t: é o **evento** onde o protocolo falha após t rounds.

Nosso objetivo é encontrar um t onde $Pr[F_t]$ seja pequena.

- O evento F_t ocorre se e somente se algum $F[i, t]$ ocorre.
- Assim, $F_t = \bigcup_{i=1}^{n} F[i, t]$.

Quando todos os processos têm sucesso?

Observação

- Antes, tínhamos considerado intersecção de eventos independentes.
- Agora, temos que considerar união de eventos dependentes.
- Calcular a probabilidade de eventos dependentes é muito difícil.
- Às vezes, é suficiente analisar usando um **limitante mais folgado**, chamado de union bound.

Quando todos os processos têm sucesso?

Union Bound

Dados eventos $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$, temos

$$Pr\left[\bigcup_{i=1}^{n} \epsilon_i\right] \leq \sum_{i=1}^{n} Pr\left[\epsilon_i\right].$$

Assim, usando o union bound, temos:

$$Pr[F_t] \leq \sum_{i=1}^{n} F[i,t] \leq n \cdot \left(1 - \frac{1}{en}\right)^t.$$

Fazendo $t = \lceil en \rceil \cdot (2 \ln n)$, a probabilidade $Pr[F_t] \leq \frac{1}{n}$.

Quando todos os processos têm sucesso?

Então, concluímos o seguinte:

Teorema

Com probabilidade pelo menos $1 - n^{-1}$*, todos processos têm sucesso ao acessar o banco de dados pelo menos uma vez em* $t = 2\lceil en \rceil \ln n$ *passos.*