Algoritmos Probabilísticos

Gilson Evandro Fortunato Dias
José Coelho de Pina (Orientador)
Instituto de Matemática e Estatística, Universidade de São Paulo

Quando falamos de algoritmos probabilísticos, estamos a falar de algoritmos que fazem escolhas aleatórias durante a sua execução. A escolha aleatória do pivô, no **Quicksort-Aleatorizado**, é um exemplo deste tipo de algoritmos.
Para estudarmos melhor a complexidade e o comportamento destes algoritmos, diante das escolhas aleatórias que os mesmos fazem, usamos o método da **Análise Probabilística**, quando a entrada do problema está bem definida num espaço de probabilidade.
O teste de primalidade[1], bastante usado em **Criptografia**, e o método de **Monte Carlo** são alguns exemplos de aplicação de algoritmos probabilísticos.

Álbum de Figurinhas

Uma das coisas mais bacanas da infância, para um bom apreciador do futebol, era o esforço para completar um álbum de figurinhas. Vamos assumir que existem n diferentes figurinhas, distribuídas independente e uniformemente, e que cada pacote contém apenas uma figurinha.

E a pergunta que seria natural fazermos: qual é o número esperado de pacotes que precisamos comprar, para completar o álbum de figurinhas do nosso *time*, sabendo que cada *time* tem n jogadores? Seja X o número de pacotes comprados até obtermos todas as figurinhas. Se X_i é o número de pacotes comprados enquanto temos $i-1$ figurinhas diferentes, e cada X_i é uma variável aleatória com distribuição geométrica, então $X = \sum_{i=1}^{n} X_i$. E quando temos $i-1$ figurinhas, a probabilidade de obtermos uma nova figurinha é

$$p_i = 1 - \frac{i-1}{n} \tag{1}$$

e portanto,

$$E[X_i] = \frac{1}{p_i} = \frac{n}{n-i+1} \tag{2}$$

O número esperado de pacotes que precisamos comprar para obter todas as figurinhas, é portanto

$$E[X] = E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} \frac{n}{n-i+1} = n\sum_{i=1}^{n} \frac{1}{i}. \tag{3}$$

Onde a soma $\sum_{i=1}^{n} \frac{1}{i}$ é conhecida como o número harmônico $H(n)$, e $H(n) = \ln n + \Theta(1)$. E portanto,

$$E[X] = n\ln n + \Theta(n) = \Theta(n \ln n). \tag{4}$$

[1] O algoritmo AKS, 2002, faz testes de primalidade determinísticos de complexidade polinomial.

Ciclos Hamiltonianos em Grafos Aleatórios

Dado um grafo G, decidir se G tem um ciclo hamiltoniano é o problema que pretendemos resolver. Lembrando que este é um problema **NP-difícil**.

Considere o modelo $G_{n,N}$ com n vértices e exatamente N arestas. Um jeito de gerarmos um grafo G uniformemente de $G_{n,N}$ é: começar sem nenhuma aresta. Escolhemos então uma das $\binom{n}{2}$ possíveis arestas aleatoriamente e adicionamos a aresta no grafo G. Agora escolhemos outra aresta das $\binom{n}{2} - 1$ possíveis arestas independente e uniformemente e voltamos a adicionar no grafo G. Continuamos a escolher até que tenhamos N arestas no grafo.

Seja G o grafo gerado em $G_{n,N}$ e assuma que $P = v_1, v_2, \ldots, v_k$ é um caminho em G e que (v_k, v_i) é uma aresta de G. Então $P' = v_1, v_2, \ldots, v_i, v_k, v_{k-1}, \ldots, v_{i+2}, v_{i+1}$ é também um caminho, e dizemos que P' é uma **rotação** de P.

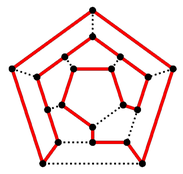

Fig. 1: Ciclo Hamiltoniano.

Algoritmo Ciclo Hamiltoniano em Grafos Aleatórios:

1. **começa** com um vértice aleatório e marque como o término do caminho.
2. **repita** até achar um ciclo hamiltoniano ou a lista de arestas não usadas, onde umas das pontas é o término do caminho estiver vazia.
3. **seja** $P = v_1, \ldots, v_k$ o caminho atual, onde v_k é o término e adiciona a aresta (v_k, u) à lista de arestas do término.
4. **remova** (v_k, u) da lista de arestas do término e da lista de arestas de u.
5. **se** $u \neq v_i$, para $1 \leq i \leq k$.
6. **então** adiciona $u = v_{k+1}$ em P e agora u é o término de P.
7. **senão**, se $u = v_i$, faça a rotação do caminho P com a aresta(v_k, v_i) e defina v_{i+1} como o término.
8. **devolva** um ciclo hamiltoniano se algum foi encontrado, ou falso se não encontrou nenhum.

Com uma pequena modificação neste algoritmo, para evitar que a distribuição das restantes arestas nas listas das arestas seja condicionada pela anterior, a análise torna-se mais fácil e o novo algoritmo modificado[1] encontra um ciclo hamiltoniano em $O(n \ln n)$ iterações com probabilidade $1 - O(n^{-1})$.

Passeios Aleatórios em Grafos

Um **processo estocástico** $X = \{X(t) : t \in T\}$ é uma coleção de variavéis aleatórias, onde t representa o tempo. Se T é um conjunto enumerável entao T é um processo de tempo discreto.[2]
Um processo estocástico de tempo discreto X_0, X_1, X_2,... é uma **cadeia de Markov** se $\Pr(X_t = a_t | X_{t-1} = a_{t-1}, X_{t-2} = a_{t-2}, ..., X_0 = a_0) = \Pr(X_t = a_t | X_{t-1} = a_{t-1})$. Ou seja, o valor de X_t depende do valor de X_{t-1}, mas não da sequência de estados que levaram o sistema a esse valor.[1]

Seja $G = (V, E)$ um grafo finito e conexo, onde $n = |V|$ e $m = |E|$. Um **caminho** num grafo é uma sequência de vértices com a seguinte propriedade: se v e w são vértices consecutivos na sequência então vw é um arco.
Dados s e t em $V(G)$, consideramos o problema de decidir se existe ou não um st-caminho.
Um **passeio aleatório** em G é uma cadeia de Markov definida pela sequência de movimentos de uma partícula P entre os vértices de G. O estado deste processo é definido pelo lugar onde a partícula se encontra num certo momento.
No estado **i**, a probabilidade da partícula P seguir pela aresta ij é $\Pr(P = j) = \frac{1}{d(i)}$, onde $d(i)$ = número de arestas que saem do vértice i.

O problema de verificar se existe um caminho entre dois vértices de G é fácil de se resolver, bastando usar uma busca em **largura** ou em **profundidade**, e tais algoritmos consomem espaço $\Omega(n)$. Vamos apresentar agora um algoritmo aleatório que trabalha apenas com $O(\log n)$ bits da memória e retorna *sim* se existe o caminho em até $4n^3$ passos, *não* caso contrário,

Algoritmo st-Caminho:

1. **Começa** o passeio aleatório em s.
2. **Se** o *passeio* atingir t em até $4n^3$ passos
3. **então** devolva *sim*
4. **senão**, devolva *não*.

O algoritmo do ***st*-caminho** com passeios aleatórios em G retorna a resposta certa quando não existe o st-caminho, e erra a resposta se ele não achar um st-caminho em até $4n^3$ passos, no passeio aleatório que ele faz em G.
Seja X o número de passos dados durante o passeio aleatório, então a probabilidade de o algoritmo retornar a resposta correta é $\frac{1}{2}$.

References

[1] M. Mitzenmacher e E. Upfal, Probability and Computing, *Cambridge*, 2005.
[2] L. Breiman, Probability and Stochastic Processes, *Mifflin*, 1969.
[3] www.ime.usp.br/∼ pf.