Algoritmos Probabilísticos

Departamento de Ciência da Computação – IME/USP

Segundo Semestre de 2006

1. Introdução

O uso de aleatoriedade em um algoritmo ou estrutura de dados pode ter diversas serventias.

Um exemplo de estrutura de dados que utiliza aleatoriedade de uma maneira simples e elegante são as *skip lists*. As skip lists incluem as mesmas operações que as árvores balanceadas de busca — busca, inserção e remoção, entre outras, porém têm uma implementação bem mais simples. O consumo de tempo esperado para tais operações em skip lists é o mesmo que o consumo de pior caso destas para árvores balanceadas de busca.

Em algoritmos, o uso de aleatoriedade pode permitir o projeto de algoritmos mais simples, pode servir para melhorar o consumo de tempo de um algoritmo, ou pode simplesmente servir de idéia inicial para a obtenção de um algoritmo determinístico para o problema. Abaixo destacamos algumas das maneiras de se usar aleatoriedade no projeto de algoritmos.

- **Embaralhamento da entrada:** o tempo consumido por um algoritmo pode depender da ordem em que os dados aparecem na entrada, como no caso do quicksort, por exemplo. Nestes casos, embaralhar a entrada pode fazer com que o tempo consumido pelo algoritmo não dependa mais da entrada, mas apenas das escolhas aleatórias feitas por ele. Como isso, algumas vezes pode-se obter uma variante do algoritmo que tem tempo esperado substancialmente melhor do que o tempo de pior caso, como acontece com o quicksort probabilístico.
- **Abundância de testemunhas:** existem muitos problemas onde se quer responder uma pergunta do tipo "A entrada tem a propriedade P?". Por exemplo, o número n é composto? Para muitas propriedades interessantes, existem objetos que servem para comprovar que a entrada tem a propriedade desejada. Em diversos casos, encontrar deterministicamente uma tal testemunha pode ser difícil, mas aleatoriedade pode ajudar. Se há um espaço de probabilidades onde as testemunhas são abundantes, então um algoritmo probabilístico pode encontrar uma tal testemunha gerando repetidamente amostras deste espaço. Se uma das amostras servir como testemunha, tem-se uma prova matemática de que a entrada tem a propriedade. Do contrário, tem-se uma forte evidência (mas nenhuma prova) de que a entrada não tem a tal propriedade. Um tal algoritmo pode portanto dar uma resposta errada. Tais algoritmos são chamados de *Monte Carlo*, enquanto que algoritmos probabilísticos que sempre produzem uma resposta correta são chamados de *Las Vegas*. O quicksort probabilístico é um exemplo de algoritmo Las Vegas enquanto que, por exemplo, a maioria dos algoritmos probabilísticos para teste de primalidade são algoritmos Monte Carlo.
- **Confundir o adversário:** ao projetar-se um algoritmo para um problema on-line, é importante controlar a razão de competitividade do algoritmo. Esta tarefa é como um jogo entre o projetista do algoritmo e um adversário, e escolhas aleatórias no algoritmo muitas vezes dificultam a vida do adversário, no sentido de que torna-se mais difícil obter uma entrada ruim para um algoritmo probabilístico.
- **Impressões digitais:** em aplicações envolvendo objetos longos, como por exemplo, longas cadeias de caracteres, pode-se representar tais objetos por "impressões digitais" curtas. Uma impressão digital pode ser um valor numérico atribuído a cada objeto por uma função de espalhamento, por exemplo. O importante é que dois objetos distintos tenham pouca chance de ter a mesma impressão digital, de maneira que, para decidir se dois tais objetos são iguais, possamos comparar não os objetos, mas suas impressões digitais. Tal estratégia pode dar origem a algoritmos mais rápidos. Um exemplo de algoritmo que segue esta linha é o algoritmo de Karp e Rabin para busca de padrão.
- **Quebra de simetria:** em computação distribuída, a quebra de deadlock é um problema importante. Para resolver tal problema e outros semelhantes é muitas vezes necessário que uma coleção de processadores cheguem a um consenso. Aleatoriedade é uma ferramenta poderosa que pode ser usada para ajudar nessa situação.

- **Rapidly mixing Markov chains:** vários algoritmos de aproximação para problemas de contagem de objetos combinatórios (por exemplo, determinar o número de emparelhamentos ou árvores em um grafo) utilizam uma estratégia de amostragem. Muitas vezes gerar uma amostra no espaço em questão é difícil por não sabermos o tamanho do espaço (que é exatamente o que queremos determinar). Em alguns casos, conseguimos gerar amostras do espaço através de cadeias de Markov. Para que tal método seja eficiente, é preciso que um passeio aleatório derivado da cadeia de Markov rapidamente esteja em um elemento qualquer da população de elementos de interesse com probabilidade uniforme. Cadeias de Markov com essa propriedade são chamadas de "rapidly mixing".

Aleatoriedade pode ainda ser usada para se provar a existência de um algoritmo para um determinado problema, sem no entanto nos dar a menor idéia de como é este algoritmo.

Vale mencionar também alguns resultados de teoria de complexidade, derivados dos sistemas interativos de prova e das provas verificáveis probabilísticamente, que evidenciam o surpreendente poder advindo do uso de aleatoriedade.

2. Objetivos e método

O objetivo desta disciplina é dar ao aluno uma idéia das várias técnicas de projeto e análise de algoritmos probabilísticos. Além disso, a disciplina deve mostrar a variedade de áreas em que tais algoritmos podem ser usados com sucesso.

Nas aulas, serão apresentadas várias técnicas de projeto e análise de algoritmos probabilísticos. Após cada aula, será recomendada a leitura de algum material bibliográfico relacionado e serão disponibilizados também alguns exercícios. É essencial que os alunos resolvam esses exercícios e leiam o material sugerido para garantir a assimilação gradativa do conteúdo apresentado nas aulas. As eventuais provas complementarão a avaliação e darão uma noção mais concreta ao professor do grau de aprendizado dos alunos.

3. Critério de avaliação

Serão disponibilizadas listas de exercícios durante o semestre. Além disso, talvez tenhamos uma ou duas provas durante o semestre. Se tivermos duas provas, elas serão nas seguintes datas: 27 de setembro e 29 de novembro.

Eventualmente, alguma outra atividade, como um seminário ou um exame oral, poderá ser também parte da avaliação.

4. Bibliografia

Os seguintes textos serão usados neste curso:

[1] M. Mitzenmachen e E. Upfal, *Probability and Computing – Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, 2005.

[2] R. Motwani e P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995. [QA845 M922r]

[3] M.X. Goemans, Lecture Notes on Randomized Algorithms, 1994. Disponíveis a partir de `http://www-math.mit.edu/~goemans/#notes`.

[4] R.M. Karp, An introduction to randomized algorithms, *Discrete Applied Mathematics* **34** (1991), 165–201.

No decorrer do semestre, provavelmente vou adicionar outros itens a esta bibliografia.

5. Outras informações

Várias informações sobre a disciplina estarão disponíveis na página

`http://www.ime.usp.br/~cris/rand/`

Vocês podem me encontrar na sala 107-C do IME-USP e o meu e-mail é `cris@ime.usp.br`.