MAC323 EXERCÍCIO-PROGRAMA 1

O PROBLEMA DA CONEXIDADE: TESTES COMPUTACIONAIS

Y. KOHAYAKAWA

Data de entrega: 24/3/2008

Introdução. Este EP supõe que você está familiarizado com o *problema da conexidade*, discutido nas primeiras duas aulas desta disciplina e as Seções 1.2 e 1.3 de [1].

Neste EP, você fará experimentos computacionais para estudar quando ocorre "conexidade total", se temos como entrada uma seqüência de pares aleatoriamente gerados.

Descrição do problema. No que segue, supomos que a instância para o problema da conexidade consiste dos pares não ordenados $e_1 = \{x_1, y_1\}$, $e_2 = \{x_2, y_2\}$, etc., onde os x_i e y_i pertencem a $\{0, 1, \ldots, N-1\}$. Lembre que, inicialmente, temos N componentes conexos e, conforme os e_i são processados, os componentes passam a se unir, formando componentes cada vez maiores. Nesse processo, o número de componentes diminui e, possivelmente, passamos a ter um único componente com N vértices. O nosso interesse é entender *quando passamos a ter um único componente, se temos uma entrada aleatória.*

Precisamos ser mais precisos sobre o que entedemos por "entrada aleatória". Experimentaremos duas possibilidades: (a) os pares $e_1, e_2, \ldots$ são gerados uniformemente ao acaso, de forma independente; (b) os pares $e_1, e_2, \ldots$ são gerados uniformemente ao acaso, de forma independente, mas com a restrição de que não há repetição, isto é, o par e_i é escolhido uniformemente ao acaso dentre todos os possíveis pares, obedecendo-se a restrição de que $e_i \neq e_j$ para todo $j < i$. É costume dizer que, em (a), os pares são gerados com reposição e, em (b), os pares são gerados sem reposição.

Estamos interessados em investigar τ tal que, com $e_1, \ldots, e_{\tau-1}$, temos mais de um componente, enquanto que com $e_1, \ldots, e_\tau$, temos um único componente; assim, τ é o momento em que ocorre "conexidade total" dos N vértices. Se não ocorre tal "conexidade total" para uma dada instância, a nossa convenção é que $\tau = -\infty$.

Outro momento de nosso interesse é aquele em que "vértices isolados", isto é, componentes com um único elemento, desaparecem. Seja σ tal que há um vértice isolado quando consideramos a seqüência $e_1, \ldots, e_{\sigma-1}$, mas não temos vértices isolados se consideramos $e_1, \ldots, e_\sigma$. Note que sempre temos $\sigma \leq \tau$. Estamos particularmente interessados em saber se a igualdade $\sigma = \tau$ ocorre com freqüência.

O que o seu programa deve fazer. O seu programa deve receber opções na linha de comando, no estilo Unix. Por exemplo, a chamada de seu programa na forma

```
prompt > ep1 -n10 -a -s271 -tau
```

implica que $N = 10$, a forma de geração aleatória dos pares é *com reposição* (forma (a) acima), a semente para o gerador de números aleatórios é 271, e a saída deve ser (unicamente) o valor de τ.

A opção `-a` acima pode naturalmente ser substituída por `-b`, para indicar que queremos geração *sem reposição* (forma (b)) acima. Note que queremos que seu programa se comporte exatamente da mesma forma toda vez que ele for chamado com os mesmos parâmetros (em particular, com a mesma semente). A opção `-tau` pode ser substituída pela opção `-f`; neste caso, queremos que seu programa tenha como saída os pares e_i para os quais ocorre união de componentes (isto é, aqueles $e_i = \{x_i, y_i\}$ para os quais x_i e y_i pertencem a componentes distintos, e e_i causa a união destes componentes). Quando o usuário especifica a opção `-tau`, ele também pode especificar a opção `-sigma`. Com ambas as opções, seu programa deve ter como saída tanto o valor de τ como o de σ.

Você também deve implementar a opção `-stats`. Quando seu programa é chamado com a opção, por exemplo, `-stats100`, ele deve executar 100 vezes o experimento de gerar os e_i e determinar τ. A saída deve ser a *média* e o *desvio padrão* dos 100 valores de τ encontrados. Neste caso, o usuário deve dar a opção `-a` ou `-b`, para especificar se cada seqüência dos e_i deve ser gerada com ou sem reposição. Ademais, a combinação de opções `-stats100 -M` indica que o experimento deve ser executado 100 vezes, mas a saída deve ser a média e o desvio padrão de $2\tau/(N \log N)$. A opção `-M` significa "mistério" (por que queremos a média e o desvio padrão desses valores?).

Para verificar o quão comum é ocorrer a identidade $\sigma = \tau$, seu programa deve aceitar a opção `-I`. Por exemplo, com a opção `-stats100 -I`, seu programa deve gerar 100 seqüências e_i e deve ter como saída o número de vezes que $\sigma = \tau$ ocorreu, juntamente com o percentual de ocorrêcia.

Uma opção que você deve implementar, útil para depuração, é a opção `-i`, que é uma alternativa às opções `-a` e `-b`. Com a opção `-i`, seu programa deve ler os pares e_i do `stdin`. A entrada deve ser dada como a seqüência de inteiros $x_1, y_1, x_2, y_2, \ldots$, com os inteiros separados por espaço (ou mudança de linha), por exemplo:

```
3 5
7 2
9 5
...
```

A combinação de opções `-F -m`M, onde M é um número natural, indica que seu programa deve gerar M pares e_i aleatórios (com ou sem reposição, dependendo da opção `-a` ou `-b` especificada pelo usuário). Seu programa deve então simplesmente imprimir os M pares gerados e, no caso de a geração ser *com reposição*, seu programa deve também ter como saída a lista dos pares que ocorreram mais de uma vez na seqüência gerada. Esta combinação de opções será útil para depuração.

Valores default. Para a conveniência do usuário, você deve supor valores *default* para as várias opções. Por exemplo, se o usuário não der a opção `-n`, seu programa deve supor que $N = 10$. A seguir, enumeramos outras convenções a serem adotadas: (i) o valor *default* para a semente é 0, (ii) entre as opções `-i`, `-a` e `-b`, o *default* é `-a`, (iii) entre as opções `-tau`, `-f`, `-stats`, o *default* é `-tau`, (iv) caso a opção `-F` seja dada, o valor *default* de M é 10.

Observações

1. *Este EP é estritamente individual.* Programas semelhantes receberão nota 0.

2. Seja cuidadoso com sua programação (correção, documentação, apresentação, clareza do código, etc), dando especial atenção a suas estruturas de dados. A correção será feita levando isso em conta.
3. Comparem entre vocês o desempenho de seus programas.
4. Entregue seu EP através do sistema Paca.
5. Não deixe de incluir em seu código um *relatório* para discutir seu EP: discuta as estruturas de dados usadas, os algoritmos usados, etc. *Se você escrever claramente como funciona seu EP, o monitor terá pouca dificuldade em corrigi-lo, e assim você terá uma nota mais alta.* (Se o monitor sofrer para entender seu código, você pode imaginar o humor dele ao atribuir sua nota.)

Observação final. Enviem dúvidas para a lista de discussão da disciplina.

Referências

[1] R. Sedgewick. *Algorithms in C: Parts 1-4, Fundamentals, Data Structures, Sorting, and Searching.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

Instituto de Matemática e Estatística, Universidade de São Paulo, Rua do Matão 1010, 05508–090 São Paulo, SP

Endereço Eletrônico: `yoshi@ime.usp.br`