CCM128 EXERCÍCIO-PROGRAMA 2

RETÂNGULOS VAZIOS

Y. KOHAYAKAWA

Data de entrega: 12/5/2014 (23:55)

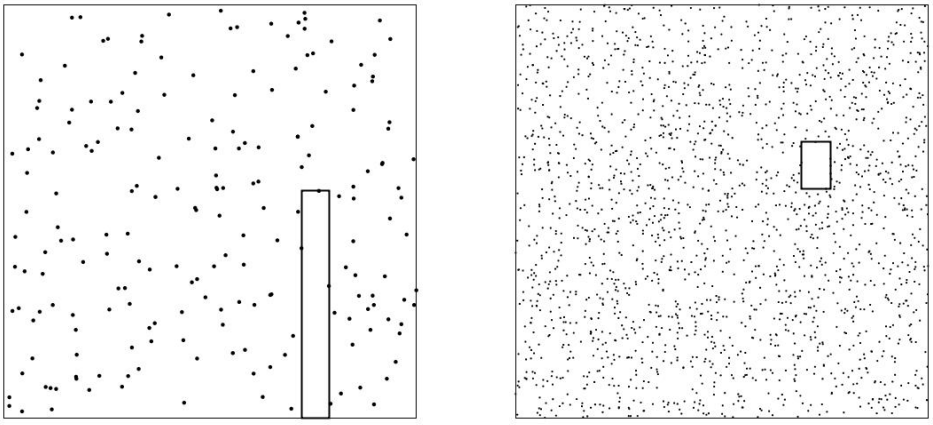
1. Introdução e definições básicas

Este EP trata de um problema de geometria computacional. Consideremos conjuntos finitos S de pontos do quadrado unitário $I^2 = I \times I = [0,1] \times [0,1]$ e 'retângulos abertos' R contidos em I^2, isto é, conjuntos da forma

$$R(x_1, x_2, y_1, y_2) = (x_1, x_2) \times (y_1, y_2) = \{(x, y) \in I^2 \colon x_1 < x < x_2 \text{ e } y_1 < y < y_2\}, \qquad (1)$$

onde x_1, x_2, y_1 e $y_2 \in I$. Dizemos que $R = R(x_1, x_2, y_1, y_2)$ *evita* S se $R \cap S = \emptyset$; também dizemos que tal R é 'vazio' (de pontos de S). O problema que queremos resolver é o problema do *maior retângulo vazio* (MRV): *dado S, encontrar um retângulo R de maior área possível que evita S.*

A Figura 1 mostra duas instâncias resolvidas.



(a) $|S| = 200$; área do retângulo vazio: 0.03640834823302188

(b) $|S| = 2000$; área do retângulo vazio: 0.007891822017010351

Figura 1. Duas instâncias resolvidas

2. Objetivo principal

Neste EP, você deve escrever um programa 'eficiente' que resolve o MRV para conjuntos $S \subset I^2$ dados. Estamos particularmente interessados no caso em que S é gerado *aleatoriamente*, com cada elemento de S escolhido ao acaso de forma uniforme em I^2, independentemente de todos

Versão de 19 de abril de 2014, 16:16.

os outros pontos de S. (Veja uma breve discussão sobre instâncias aleatórias na Seção 3.5.) Seu programa deve ser capaz de resolver tais instâncias S com centenas ou milhares de pontos rapidamente (esta exigência está formulada de forma vaga propositadamente).

3. Seu programa

É natural decompor seu programa em vários módulos. Por exemplo, é natural implementar classes para pontos, conjuntos de pontos e retângulos.

Ademais, você deve implementar um módulo de visualização, que exibe a instância e a solução encontrada (isto é, um módulo que gera figuras como aquelas na Figura 1).

Você deve também implementar um módulo para executar o *doubling test*. Você deve usar este módulo para estimar a complexidade de tempo de seu algoritmo para resolver o MRV em instâncias aleatórias. Não deixe de discutir sua conclusão (empírica) sobre a complexidade de seu algoritmo em seu relatório. Procure também fazer uma análise teórica da complexidade de seu algoritmo.

Você deve também implementar um módulo que, dado N, estima o valor esperado da área do maior retângulo vazio relativo a um conjunto aleatório S com $|S| = N$. Finalmente, você deve implementar um módulo que gera um gráfico deste valor esperado em função de N.

No que segue, discutimos os módulos que você deve implementar, dando exemplos de seus usos (estes exemplos devem tornar mais claro o que é pedido neste EP).

3.1. `Point.java`, `PointSet.java`, `Rectangle.java`. Essas classes devem implementar os tipos `Point`, `PointSet` e `Rectangle`. Implemente as seguintes formas de execução de `PointSet`. Para gerar um conjunto aleatório com N pontos ($N = 6$):

```
$ java-introcs PointSet -r 6
6
0.19079518105227067 0.7781257632978256
0.4974153326919968 0.2503665625736603
0.5213656070164971 0.38069973596554463
0.22754335354306854 0.6710899050769443
0.1754296870531391 0.36021400982688445
0.3817382183726922 0.3955423012618332
```

O conjunto gerado pode ser armazenado para ser processado posteriormente:

```
$ java-introcs PointSet -r 6 > data.6
$ cat data.6
6
0.36669290177551617 0.4717504294232526
0.5483975883724658 0.5076621962517646
0.9544398399611626 0.3500417766273609
0.006611416732648867 0.235248480395164
0.4078041440144815 0.3870395237696017
0.07633625320490867 0.5961843373135058
$ java-introcs PointSet -i < data.6
Max area: 0.4547545804989615
```

3.2. `Visualize.java`. Para resolver uma dada instância e fazer a ilustração da solução:

```
$ java-introcs Visualize -i < data.6
```

A execução acima deve gerar uma saída como na Figura 2. Para gerar instâncias aleatórias

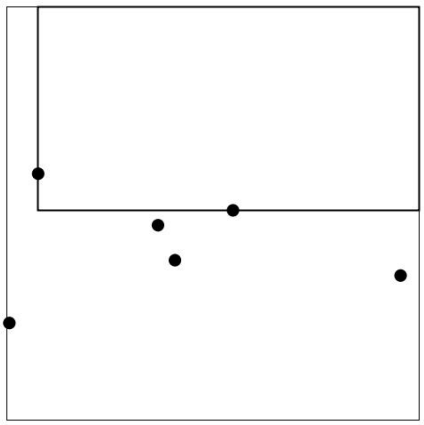


FIGURA 2. Solução para o conjunto em `data.6`

com N pontos M vezes e exibir as soluções graficamente ($N = 800$ e $M = 10$):

```
$ java-introcs Visualize 800 10
```

Para armazenar as figuras geradas por `Visualize`, você deve permitir o usuário a executar

```
$ java-introcs Visualize -i data-500 < data.500
```

Nesta execução, devem ser gerados os arquivos `data-500.jpg` e `data-500.txt`, com o primeiro contendo uma figura ilustrando a solução e o segundo contendo a saída em formato texto (você pode decidir quais informações colocar neste segundo arquivo, mas a área do retângulo encontrado deve fazer parte). Implemente também o seguinte formato de execução:

```
$ java-introcs Visualize 1000 3 random-1000
```

Neste caso, devem ser gerados 6 arquivos (três com figuras e três com a saída em formato texto).

3.3. `DoublingTest.java`. Para executar o *doubling test* para estimar a complexidade de seu algoritmo em instâncias aleatórias, o usuário deve poder fazer

```
$ java-introcs DoublingTest
     N    time ratio
    16    0.00 2.00
    32    0.01 4.50
    64    0.05 5.78
   128    0.20 3.79
  [...]
```

3.4. `AveArea.java` e `AreaPlot.java`. Para estimar o valor esperado da área do maior retângulo vazio para instâncias aleatórias com N pontos, o usuário poderá gerar tais instâncias M vezes e calcular a média ($N = 500$ e $M = 100$):

```
$ java-introcs AveArea 500 100
Average = 0.023699853652082905
```

Por conveniência, denotemos por $\mu(N)$ o valor esperado do maior retângulo vazio para instâncias aleatórias com N pontos. Lembre a técnica usada em `PercPlot.java` para gerar gráficos. Use aquela técnica para gerar gráficos como na Figura 3. O gráfico na Figura 3 foi obtido executando-se

```
$ java-introcs AreaPlot 500
```

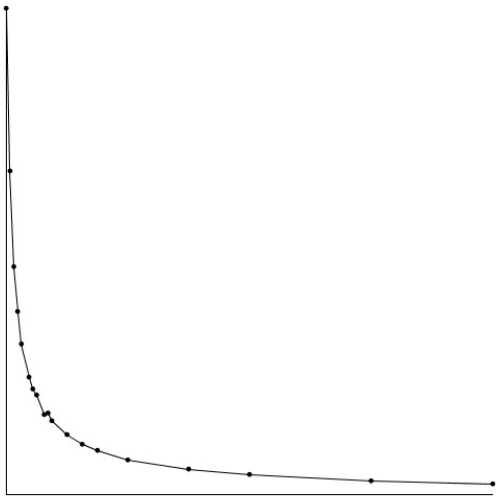


FIGURA 3. Uma aproximação (experimental) do gráfico de $\mu(N)$

Neste gráfico, o eixo x corresponde aos valores de N, com $0 \leq N \leq 500$ (`AreaPlot` foi executado com $N = 500$), enquanto que o eixo y corresponde aos valores (estimados) de $\mu(N)$. O ponto mais à esquerda no gráfico (que é $(0, 1)$) corresponde ao caso $N = 0$. O ponto mais à direita (que é algo como $(1, 0.0236\ldots)$) corresponde ao caso $N = 500$.

Usando seu programa `AreaPlot` ou programas semelhantes, você consegue conjecturar como a função $\mu(N)$ decai conforme N cresce?

3.5. **Instâncias aleatórias.** Seu EP deve ser capaz de resolver instâncias dadas (a serem lidas no `stdin`), mas também deve ser capaz de gerar e resolver instâncias aleatórias. As instâncias aleatórias devem ser geradas da seguinte forma: dado N, você deve gerar P_i $(0 \leq i < N)$ uniformemente ao acaso no quadrado I^2, usando `StdRandom.uniform()` (este método gera quantidades `double` em $[0, 1)$ e não em $I = [0, 1]$, mas você pode ignorar este fato). Neste enunciado, quando falamos de *instâncias aleatórias*, estamos sempre falando de instâncias geradas dessa forma.

4. DESAFIO (OPCIONAL)

Resolva o problema do *maior disco vazio* (MDV): *dado S, encontrar um disco aberto D de maior área possível que evita S*. Por simplicidade, neste problema, você deve entender que $S \subset I^2$ corresponde à instância (infinita) $\mathbb{Z}^2 + S = \{(i, j) + P \colon i, j \in \mathbb{Z} \text{ e } P \in S\}$ e que estamos à procura de um disco $D \subset \mathbb{R}^2$ de área máxima que evita S (nesta versão, a fronteira do quadrado unitário não impõe restrições).

Este desafio pode ser entregue até o final do semestre. Se você pretende fazer este desafio, escreva para o monitor e para mim, para que saibamos que você está trabalhando nele.

Observações. Seguem algumas observações importantes.

1. *Este EP é estritamente individual.* Programas semelhantes receberão nota 0.
2. Seja cuidadoso com sua programação (correção, documentação, apresentação, clareza de código, etc). A correção não é baseada apenas na correção de seu programa.
3. Comparem entre vocês o desempenho de seus programas.
4. Entregue seu EP no Paca.

5. Não deixe de incluir em seu material um *relatório* para discutir seu EP: faça quaisquer comentários que você achar interessante e inclua exemplos de execuções de seu programa.

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

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

Endereço eletrônico: yoshi@ime.usp.br