Lista 2 - MAC0122 Princípios de Desenvolvimento de Algoritmos - POLI

Prof. Ronaldo Fumio Hashimoto

1 Exercícios

Os exercícios de 11 a 23 correspondem, respectivamente, aos exercícios de 3.11 a 3.23 do livro de Robert Sedgewick [1].

11. Suponha que seja declarado um `int a[99]`. Mostre o conteúdo do vetor depois que as seguintes linhas de códigos forem executadas.

```
for (i = 0; i < 99; i++) a[i] = 98-i;
for (i = 0; i < 99; i++) a[i] = a[a[i]];
```

12. Modifique a implementação de crivo de Eratóstenes abaixo (Programa 3.5 do livro) para usar um vetor de (*i*) `chars`; e (*ii*) `bits`. Determine o efeito dessas mudanças na quantidade de espaço e tempo utilizado para o programa.

    Programa 3.5

```
#define N 10000
main()
   { int i, j, a[N];
     for (i = 2; i < N; i++) a[i] = 1;
     for (i = 2; i < N; i++)
        if (a[i])
           for (j = i; i*j < N; j++) a[i*j] = 0;
     for (i = 2; i < N; i++)
        if (a[i]) printf("%4d ", i);
     printf("\n");
   }
```

13. Uso o crivo de Eratóstenes para determinar o número de primos menores que N, para $N = 10^3$, 10^4, 10^5, e 10^6.

14. Use o crivo de Eratóstenes para desenhar uma plotagem de N versus o número de primos menores que N para N entre 1 e 1000.

15. Determine empiricamente o efeito de remover o teste `if (a[i])` que cobre o *loop* interno do Programa do exercício 12 (Programa 3.5 do livro), para $N = 10^3$, 10^4, 10^5, e 10^6.

16. Analise o programa do exercício 12 (Programa 3.5 do livro) para explicar o efeito que você observou no Exercício 15.

17. Escreva um programa que conta o número de inteiros diferentes menor que 1000 que aparece no fluxo da entrada.

18. Escreva um programa que determina empiricamente o número de inteiros positivos aleatórios menores que 1000 que é esperado antes que seja gerado um número repetido.

19. Escreva um programa que determina empiricamente o número de inteiros positivos aleatórios menores que 1000 esperado para que seja gerado cada valor pelo menos uma vez.

20. Modifique o programa abaixo (Programa 3.7 do livro) para simular a situação que o lançamento de uma moeda resulta em cara com probabilidade p. Rode 1000 experimentos, e em cada experimento simule 32 lançamentos de moeda com $p = 1/6$ compare os resultados obtidos com a figura abaixo (Figura 3.2 do livro).

```
#include <stdlib.h>
int heads()
   { return rand() < RAND_MAX/2; }
main(int argc, char *argv[])
  { int i, j, cnt;
    int N = atoi(argv[1]), M = atoi(argv[2]);
    int *f = malloc((N+1)*sizeof(int));
    for (j = 0; j <= N; j++) f[j] = 0;
    for (i = 0; i < M; i++, f[cnt]++)
       for (cnt = 0, j = 0; j <= N; j++)
         if (heads()) cnt++;
    for (j = 0; j <= N; j++)
      {
         printf("%2d ", j);
         for (i = 0; i < f[j]; i+=10) printf("*");
         printf("\n");
      }
   }
```

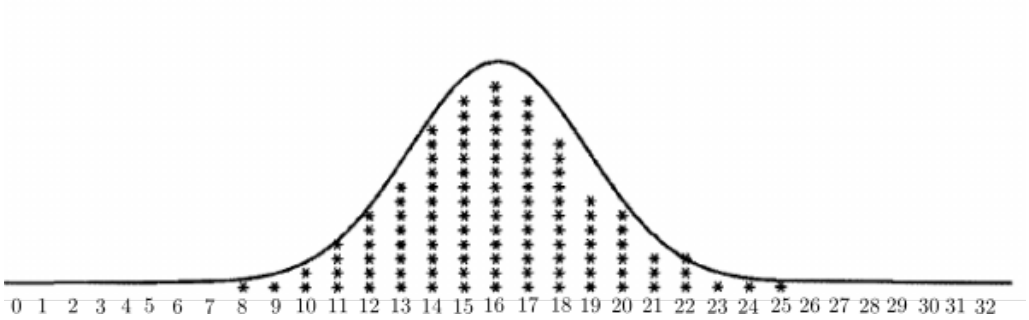

Figura 1: Figura adaptada de [2].
Simulação de lançamento de moedas
Esta figura exibe o resultado da execução do programa acima (Programa 3.7), simulando 1000 experimentos de 32 lançamentos de moedas. O número de caras que devemos ver é aproximado pela função da distribuição normal desenhada sobre os dados.

21. Modifique o programa do exercício 20 (Programa 3.7 do livro) para simular a situação que o lançamento de uma moeda resulta em cara com probabilidade λ/N. Rode 1000 experimentos, e em cada experimento simule 32 lançamentos de moeda e compare os resultados obtidos com a Figura 1 do exercício 20 (Figura 3.2 do livro). Esta distribuição é conhecida como a clássica distribuição de *Poisson*.

22. Modifique o programa abaixo (Programa 3.8 do livro) e imprima as coordenadas do par de pontos mais próximos.

    Programa 3.8

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "Point.h"
float randFloat()
   { return 1.0*rand() / RAND_MAX; }
main(int argc, char* argv[])
   { float d = atof(argv[2]);
     int i, j, cnt = 0, N = atoi(argv[1]);
     point *a = malloc(N * (sizeof(*a)));
     for (i = 0; i < N; i++)
        { a[i].x = randFloat(); a[i].y = randFloat(); }
     for (i = 0; i < N; i++)
        for (j = i+1; j < N; j++)
           if (distance(a[i], a[j]) < d) cnt++;
     printf("%d edges shorter than %f\n", cnt, d);
  }
```

23. Modifique o programa do exercício 22 (Programa 3.8 do livro) para executar a mesma computação em d dimensões.

Referências

[1] R. Sedgewick, "Algorithms in c—third edition," pp. 89–90, 1998.

[2] R. Sedgewick, "Algorithms in c—third edition," p. 86, 1998.