

## Problema da conectividade

- ▷ Definição do problema; union-find
- ▷ Algoritmos quick find, quick union, quick union com pesos, [quick union com pesos e compressão de caminhos]
- ▷ Quick find, quick union: comportamento quadrático:  $\geq cN^2$  instruções para certas instâncias (arbitrariamente grandes), onde  $c$  é uma constante positiva
- ▷ Quick union com pesos: complexidade de tempo  $O(M \log N)$  para seqüências com  $M$  pares e  $N$  objetos

## Problema da conectividade

**Propriedade 1.** *Suponha que executamos o algoritmo quick find em uma instância com  $M$  pares e  $N$  objetos, e que a saída tem  $S$  pares. Então o algoritmo executou pelo menos  $NS$  instruções (por exemplo, ele executou o teste  $\text{id}[i] == t$  pelo menos este número de vezes).*

**Propriedade 2.** *O algoritmo quick union pode chegar a executar  $(N - 1)(N - 2)/2$  instruções para resolver o problema da conectividade com  $N - 1$  pares e  $N$  objetos.*

**Propriedade 3 (Custo do find).** *O algoritmo quick union com pesos segue  $\leq \log_2 N$  ponteiros para chegar à raiz da árvore que contém o elemento de partida. Isto é, find custa tempo  $O(\log N)$ .*

## Exercício

▷ **Instância conexa:** dizemos que uma instância é conexa quando a saída tem  $N - 1$  pares.

Para contar o número de pares na saída, basta fazer

```
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx
$ prog1.1 < prog1.1.in | wc -l
9
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx
$
```

Verifique experimentalmente a *probabilidade de uma instância aleatória ser conexa* para valores grandes de  $N$  e  $M$  por volta de  $N(\log N)/2$ . (Você tem de escrever um pequeno programa para gerar as instâncias aleatórias.)

## Crescimento de funções

- ▶ Contraste entre algoritmos de tempo  $N^2$  e  $N \log N$

## Para meditar

segundos

$10^2$	1.7 minutos
$10^4$	2.8 horas
$10^5$	1.1 dias
$10^6$	1.6 semanas
$10^7$	3.8 meses
$10^8$	3.1 anos
$10^9$	3.1 décadas
$10^{10}$	3.1 séculos
$10^{11}$	...

## Crescimento de funções/tempo de execução

ops/seg	$N = 10^6$			$N = 10^9$		
	N	$N \log N$	$N^2$	N	$N \log N$	$N^2$
$10^6$	segs	minutos	meses	horas	dias	nunca
$10^9$	piscar	piscar	horas	segs	minutos	séculos
$10^{12}$	piscar	piscar	segs	piscar	piscar	meses

## Notação assintótica

▷ Notação  $O$ ,  $\Omega$ ,  $\Theta$

**Definição 4.** *Seja  $f: \mathbb{N} \rightarrow \mathbb{R}$  uma função. Escrevemos  $O(f(N))$  para qualquer função  $g: \mathbb{N} \rightarrow \mathbb{R}$  para a qual existem constantes  $c > 0$  e  $N_0$  tais que  $|g(N)| \leq c|f(N)|$  para todo  $N \geq N_0$ .*

**Definição 5.** *Seja  $f: \mathbb{N} \rightarrow \mathbb{R}$  uma função. Escrevemos  $\Omega(f(N))$  para qualquer função  $g: \mathbb{N} \rightarrow \mathbb{R}$  para a qual existem constantes  $c > 0$  e  $N_0$  tais que  $|g(N)| \geq c|f(N)|$  para todo  $N \geq N_0$ .*

**Definição 6.** *Escrevemos  $\Theta(f(N))$  para qualquer função  $g(N)$  tal que  $g(N) = O(f(N))$  e  $g(N) = \Omega(f(N))$ .*

## Notação assintótica

▷ Notação  $o$

**Definição 7.** *Seja  $f: \mathbb{N} \rightarrow \mathbb{R}$  uma função. Escrevemos  $o(f(N))$  para qualquer função  $g: \mathbb{N} \rightarrow \mathbb{R}$  tal que*

$$\lim_{N \rightarrow \infty} \frac{g(N)}{f(N)} = 0. \quad (1)$$

## Exemplo de uso

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int i, j, k, n = atoi(argv[1]), t = 0;

    for (i = 0; i < n; i++)
        for (j = 0; j < i; j++)
            for (k = 0; k < j; k++)
                t++;
    printf("%d: %d\n", n, t);

    return 0;
}
```

## Exemplo de uso

Valor de  $t$ ?

▷ Resposta:  $t = \binom{n}{3} = \frac{1}{3!}n(n-1)(n-2)$

▷ Outra resposta:

$$t = \frac{1}{3!}(n^3 - 3n^2 + 2n)$$

▷ Outra resposta:  $t = n^3/6 + O(n^2)$

▷ Outra resposta:  $t = O(n^3)$

## Exemplo de uso

Valor de  $t$ ?

- ▷ Resposta:  $t = \binom{n}{3} = \frac{1}{3!}n(n-1)(n-2)$
- ▷ Outra resposta:  $t = n^3/6 + \Theta(n^2)$
- ▷ Outra resposta:  $t = \Theta(n^3)$

## Tipos primitivos

- ▷ Caracteres (`chars`)
  - ▷ Inteiros (`ints`)
  - ▷ Números em ponto flutuante (`floats`)
- ▷ **Leia as seções sobre caracteres e inteiros na página de Feofiloff**  
[<http://www.ime.usp.br/~pf/algoritmos/>]

## Tipos definidos pelo usuário

$x_1, \dots, x_N$  seqüência de  $N > 0$  números:

▷ média:  $\mu = \frac{1}{N} \sum_{1 \leq i \leq N} x_i$

▷ variância:  $\sigma^2 = \frac{1}{N} \sum_{1 \leq i \leq N} (x_i - \mu)^2$

▷ desvio padrão:  $\sigma$

Exercício:

$$\sigma^2 = \left( \frac{1}{N} \sum_{1 \leq i \leq N} x_i^2 \right) - \mu^2$$

## Tipos definidos pelo usuário

```
/* prog3.2.c */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

typedef int numType;

numType randNum()
    { return rand(); }
```

## Tipos definidos pelo usuário

```
main(int argc, char *argv[])
{
    int i, N = atoi(argv[1]);
    float m1 = 0.0, m2 = 0.0;
    numType x;
    for (i = 0; i < N; i++)
    {
        x = randNum();
        m1 += ((float) x)/N;
        m2 += ((float) x*x)/N;
    }
    printf("          Average: %f\n", m1);
    printf("Std. deviation: %f\n", sqrt(m2-m1*m1));
}
```

## Alternativa recomendada

- ▷ Interface
- ▷ Implementação
- ▷ Cliente

## Alternativa recomendada

- ▷ Interface: `Num3.2.h`
- ▷ Implementação: `Num3.2.c`
- ▷ Cliente: `prog3.2_client.c`

## Alternativa recomendada

Arquivo Num3.2.h:

```
/* Num3.2.h */  
typedef int numType;  
numType randNum();
```

Arquivo Num3.2.c:

```
/* Num3.2.c */  
#include <stdlib.h>  
#include "Num3.2.h"  
  
numType randNum()  
{ return rand(); }
```

## Alternativa recomendada

```
/* prog3.2_client.c */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <Num3.2.h>

main(int argc, char *argv[])
{ int i, N = atoi(argv[1]);
  float m1 = 0.0, m2 = 0.0;
  numType x;
```

## Alternativa recomendada

```
/* prog3.2_client.c */
  (cont.)
  for (i = 0; i < N; i++)
  {
    x = randNum();
    m1 += ((float) x)/N;
    m2 += ((float) x*x)/N;
  }
  printf("      Average: %f\n", m1);
  printf("Std. deviation: %f\n", sqrt(m2-m1*m1));
}
```

## Alternativa recomendada

```
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ gcc -g -I. -Wall -pedantic -ansi -c prog3.2_client.c Num3.2.c
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ gcc -lm prog3.2_client.o Num3.2.o -o prog3.2_client
yoshi@erdos:~/Main/www/2006ii/mac122a/exx
$ prog3.2_client 1000
    Average: 1091191424.000000
Std. deviation: 609526184.005430
yoshi@erdos:~/Main/www/2006ii/mac122a/exx$
```