Relatório: Exercício Programa 01

Lucas Sung Jun Hong (n.USP: 8124329)

13 de outubro de 2014

Sumário

1 Estruturação do Programa

1.1 Inserindo n a um vetor

Recebido **n**, um `unsigned long long int`, cada algarismo é inserido dentro de um vetor **nArray[ ]**. Seu tamanho será **x**, tal que **x** foi obtido pela função `int algCount`:

```
int algCount (unsigned long long int n) {
    int count = 0;

    while (n != 0) {
        n /= 10;
        count++;
    }
    return count;
}
```

O vetor contém na posição 0, o algarismo menos significativo e na posição x - 1, o algarismo mais significativo. Por exemplo:

`input:  n = 65; output:  5 6`

1.2 Circular Doubly Linked-List

Considerando que precisamos de números extremamente grandes, porém unidimensionais, pois trata-se de números elevados à uma potência de 2, foi optado uma lista circular duplamente encadeada. Desconhecemos o tamanho que necessitamos gerar a cada vez que um número é multiplicado por 2 e mais,precisamos atualizar os elementos constantemente. Assim, usando uma lista circular duplamente encadeada facilita aumentar algarismos extras sendo que só é preciso criar uma célula extra.

No entanto, o problema de usar uma lista encadeada é "memory overhead"pois cada célula aponta duplamente para frente e para trás.

A lista é criada da seguinte forma:

```
ListaCelula *crieListaCelula () {
    ListaCelula *lst;
    Celula *p;

    lst = malloc(sizeof(*lst));
    lst->head = malloc(sizeof(*lst->head));

    p = crieCelula();
    p->element = 2;

    lst->head->prox = p;
    lst->head->ant = p;
    p->prox = lst->head;
    p->ant = lst->head;

    return lst;
}
```

Ilustrativamente, a função faz o seguinte:

Ao dobrarmos o conteúdo de uma célula, obtemos um momento em que seu valor é maior que 9, ou melhor, teremos 2 algarismos (por exemplo o 18) dentro dessa célula. Nesse momento criamos uma nova célula e inserimos nela o algarismo mais significativo, deixando o algarismo menos significativo na céula inicial.

Figura 1: criação de uma lista

A implementação no programa foi escrita de uma forma tal que a cada criação de uma nova célula, o dígito 1 já vem incluso, sendo apenas necessário obter o resto da divisão do número (maior que 9) por 10 e substituí-la na célula inicial.

Figura 2: Após de dobrar os algarismos, checamos de trás para frente para verificar se cada célula possui número > 9. Caso sim, atualizamos cada célula com a soma correta.

2 Quando comparamos?

2.1 O que é 2^{min}?

Temos n com $0 \leq n \leq 2 * 10^9$. Com n, podemos contar quantos algarismos contém em n, sendo esse valor: x. Assim, um número mínimo que contenha o n é o seguinte: $n * 10^{x+mod}$, tal que $mod = 1, 2, 3, ...$ Calculando o logaritmo na base 2 desse número podemos obter a potência desejada, tal que, 2 elevado à esse número contenha n.

Por exemplo: seja $n = 12$ então $x = 2$, com $mod = 1$. Então o número mínimo será 12000. assim: *(com mod = 1)*

$$log_2 12000 = log_2 12 * 10^{2+1} = log_2 12 * 10^3 =$$
$$log_2 12 + 3 * (log_2 10) = 13.550$$

como $min = \lceil\ (log_2(n) + ((x + mod) * log_2(10)))\ \rceil$

$$min = \lceil log_2 12000 \rceil = \lceil 13.550 \rceil = 14$$

Assim 2^{14} será o valor estimado. Em outras palavras, esperamos que 2^{14} seja o valor ideal que contenha os dígitos "12"no início. Mas $2^{14} = 16384$, portanto 12000 não foi um bom valor. Aumentamos o $mod+1$, portanto $mod = 2$, assim teremos: $x + (mod) = 2 + (2) = 4$, ou seja, aumentamos de 12,000 para 120,000. *(com mod = 2)*

$$log_2 120000 = log_2 12 * 10^{2+2} = log_2 12 * 10^4$$

E assim aumentamos $mod + 1$ caso 2^{min} não for o valor esperado até encontrarmos o valor ideal.

2.2 Comparando 2^{pot} com 2^{min}

O programa calcula 2^{pot}, com $pot = 1, 2, 3, ...$ No momento em que $pot = min$, chama-se a função `compara(lst, nArray, x)`, que compara 2^{pot} com 2^{min}. Caso $2^{min} \neq 2^{pot}$, aumentamos o valor de min até encontrarmos o valor ideal.

3 Maior valor testado

3.1 13,7 minutos

O maior valor de n testado foi: 1,999,987,841.

Sua saída foi: 904,665.

Para tal n, o programa levou 825,97 segundos (13,7 minutos).

O número calculado foi:

4 Bibliografia

- http://en.wikipedia.org/wiki/Power_of_two
- http://en.wikipedia.org/wiki/Binary_numeral_system
- http://en.wikipedia.org/wiki/Benford's_law
- http://en.wikipedia.org/wiki/Exponentiation
- http://en.wikipedia.org/wiki/Logarithm
- http://oeis.org/A001146
- http://en.wikibooks.org/wiki/LaTeX/
- http://stackoverflow.com/
- http://en.wikipedia.org/wiki/Bignum
- http://www.cs.utexas.edu/users/djimenez/utsa/cs1723/lecture1.html
- https://gmplib.org/
- https://www.khanacademy.org/math/algebra2/logarithms-tutorial/logarithmic-scale-patterns/v/benford-s-law-explanation--sequel-to-mysteries-of-benford-s-law
- http://valgrind.org/docs/manual/mc-manual.html
- http://valgrind.org/docs/manual/manual-core.html#opt.read-var-info
- http://valgrind.org/docs/manual/manual-core-adv.html#manual-core-adv.gdbserver-simple