Sistema de Tranca Digital

Sophia Mendes da Silveira∗, NOME†

Faculdade de Informática — PUCRS 27 de março de 2023

**Resumo**

Este artigo descreve alternativas de solução para o primeiro trabalho proposto na disciplina de Sistemas Digitais no semestre 1/2024, e trata do desenvolvimento de uma Máquina de Estados Finita com o objetivo de simular uma tranca digital.

÷

A Máquina de Estados Finita construida compreende os estados da tranca, dos quais iniciam com a configuração da mesma, tentativas de aberturas, falhas na abertura que levam ao acionamento do alarme e o sucesso na abertura.

# Introdução

Neste cenário da disciplina de Sistemas Digitais, o problema apresentado basea-se em criar e simular uma Tranca Digital por meio de uma Máquina de estados Finitos. Tendo seus critérios para a configuração, leitura de entradas, correção das entradas e abertura:

* Configuração: Para esta etapa é necessário que

# Primeira solução

Depois de considerar o problema, podemos perceber que o valor inicial *a* = 1 serve apenas para fornecer um número inicial sobre o qual podem ser aplicadas as regras para gerar outros números, e portanto seu valor não tem importância especial. Com esta constatação, e sabendo que em um inteiro de 32 bits podemos armazenar números até 231 ou até 320 (obtivemos estes dados através de testes), podemos propor uma solução bastante simples: bastaria criar frações 2*i/*3 *j* com todas as possibilidades de expoentes para 2*i* (ou seja, 0 *i* 31) e para 3 *j* (ou seja, 0 *j* 20), verificando quais números são gerados. O total de números a serem testados é de apenas 32 21 = 672, portanto o algoritmo deve ser executado com grande velocidade. Um algoritmo implementando esta idéia seria parecido com este:

∗

≤ ≤ ≤ ≤

1 *registro tab*[100] {

2 *int achei* , *int exp2* , *int exp3*

3 }

4

5 **procedimento** ALGORITMO1

6 **para** *i* 1 *até* 100 *faça*

←

7 *tab*[*i*] *FALSE*

←

8 **fim**

9 **para** *i* 1 *até* 100 *faça*

←

10 **para** *j* 0 *até* 20 *faça*

←

11 *num* 2*i/*3 *j*

←

12 // *Se está na faixa desejada e ainda não encontramos*

13 **se** *num* ≥ 100 *e tab*[*num*] = *FALSE então*

14 *tab*[*num*].*achei* ← *TRUE*

15 *tab*[*num*].*exp2 i*

←

16 *tab*[*num*].*exp3 j*

←

17 **fim**

18 **fim**

19 **fim**

20 **fim**

Foi criada uma tabela para manter os resultados encontrados, e a cada resultado encontrado seus expoentes são armazenados. Ao final basta imprimir a tabela, colocando-a no formato adequado de saída (este algoritmo é simples e não será mostrado aqui).

Também é interessante perceber que a solução apresentada até o momento acha os menores ex- poentes possíveis, mas precisa manter o controle de que números já foram encontrados, para não escrever expoentes maiores sobre valores já armazenados. Uma maneira muito simples de descartar este tipo de controle é fazendo o loop externo **decrescente**, garantindo assim que os menores expo- entes serão encontrados por último, e permitindo a escrita sobre valores já encontrados. Esta solução seria:

1 *registro tab*[100] {

2 *int exp2* , *int exp3*

3 }

4

5 **procedimento** ALGORITMO2

6 **para** *i* 31 *até* 0 *faça*

←

7 **para** *j* 0 *até* 20 *faça*

←

8 *num* 2*i/*3 *j*

←

9 **se** *num* ≤ 100 *então*

10 *tab*[*num*].*exp2 i*

←

11 *tab*[*num*].*exp3 j*

←

12 **fim**

13 **fim**

14 **fim**

15 **fim**

Esta idéia é bastante simples e eficiente, porém um teste mostra que ela acha apenas 53 dos 99 números desejados. Com isto podemos concluir que os outros 46 valores devem ser obtidos com expoentes maiores, e os números envolvidos não podem ser representados em inteiros de 32 bits.

*Note que esta seção dá uma idéia inicial, analisa-a de forma rápida e diz porque ela não serve, também de forma rápida, sem perder o tempo do leitor com papo furado. Essa precisão e detalhe são sinais de um trabalho onde vem coisas mais interessantes mais tarde. Veja que não foi usada uma linguagem de programação com todos os detalhezinhos! Os algoritmos estão em uma linguagem simplificada, que permite que se entenda rapidamente o que deve ser feito, sem excesso de detalhes inúteis.*

*Perceba também que o código não é cheio de explicações óbvias e inúteis (esta linha incrementa o contador, esta linha calcula o número), mas é clara o bastante para que outra pessoa entenda o que é feito e reproduza o algoritmo em sua linguagem preferida. Uma falha: terminamos dizendo que achamos 53 dos 99 números, mas sem dizer quais foram nem falar do tempo necessário para encontrá-los.* ***Moral****: quando terminar de falar de uma solução, apresente os resultados, comente o tempo e as (possíveis) falhas!!*

# Segunda solução

Uma vez que a alternativa mais simples falha em quase metade dos casos, optamos por desenvolver uma abordagem mais cuidadosa, experimentando um truque bastante simples: se já tivermos os ex- poentes para certos números (obtidos, por exemplo, com o método anterior), podemos criar novos números a partir destes. Por exemplo, se sabemos que 2 = 21 30 e que 3 = 25 32, então podemos

÷ ÷

tentar produzir 6 através da combinação das duas representações: 6 = 2 3 = 21+5 30+2 = 26 32*.*

∗ ÷ ÷

Infelizmente, conferindo o resultado descobrimos que 26 32 = 64*/*9 = 7*.*1111 *. . .* e portanto não obtemos 6, e sim 7. Tabulando os resultados e prestando atenção nos valores, temos

÷

|  |  |  |  |
| --- | --- | --- | --- |
| Valor truncado | Fórmula | Valor exato | Excesso |
| 2 | 21 ÷ 30 | 2 | 0 |
| 3 | 25 ÷ 32 | 3.5555. . . | 0.5555. . . |
| 6 | 26 ÷ 32 | 7.1111. . . | 1.1111. . . |

O fenômeno parece bastante simples: como a representação de 3 carrega consigo um erro (ex- cesso) de 0.5555. . . , ao multiplicar 3 por 2 o erro também dobra, fazendo com que o valor obtido seja maior do que 6. Como se pode ver na tabela, o erro para 6 é de 1.1111. . . , exatamente o dobro de 0.5555. . . .

Desta maneira, podemos perceber que um mecanismo simples de geração de novos números a partir de outros que já são conhecidos teria, no mínimo, os seguintes problemas:

1. São necessários valores iniciais que são usados como “sementes”. Estes valores iniciais podem ser obtidos com o método anterior, por exemplo.
2. Os números sendo procurados devem ser decompostos para que se saiba que outros números devem ser multiplicados para obtê-los, da mesma forma que constatamos que 6 poderia ser produzido com 2 e 3. Esta decomposição é relativamente trabalhosa, e envolve números primos e vários tipos de testes.
3. É preciso conhecer a representação para os números primos, pois estes não podem ser obtidos multiplicando-se outros valores. Por exemplo, o valor 19 não poderá ser decomposto e sua representação tem que ser descoberta de forma independente.
4. É preciso ter controle do erro sendo transmitido entre os números, a fim de garantir que estamos chegando aos números corretos.

Por todos estes motivos optamos por não desenvolver esta solução, mas com o raciocínio feito até o momento podemos elaborar uma solução aparentemente mais complexa, mas que se revela mais simples quando analisada. Esta solução será apresentada a seguir.

*Veja como esta segunda solução se apoia sobre a primeira, e é descrita em um pouco mais de detalhe. Motivo: além de ser menos óbvia, ela serve de ponte para a terceira alternativa, que será a verdadeira resposta, e aqui precisamos deixar claro como funciona o mecanismo de propagação do erro. Isto é fundamental para o que vem a seguir. Note que esta solução é descartada com uma lista de motivos, para que ninguém pergunte porque ela não foi implementada.*

# Resultados

Depois de implementar o algoritmo acima em linguagem C e executá-lo num tempo de aproximada- mente meio segundo, obtivemos os seguintes resultados:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | = | 2^0 | / | 3^0 | 26 | = | 2^19 / 3^9 | | | 51 | = | 2^58 / 3^33 | | | 76 | = | 2^76 / 3^44 | | |
| 2 | = | 2^1 | / | 3^0 | 27 | = | 2^46 / 3^26 | | | 52 | = | 2^39 / 3^21 | | | 77 | = | 2^57 / 3^32 | | |
| 3 | = | 2^5 | / | 3^2 | 28 | = | 2^8 / 3^2 | | | 53 | = | 2^20 / 3^9 | | | 78 | = | 2^38 / 3^20 | | |
| 4 | = | 2^2 | / | 3^0 | 29 | = | 2^16 / 3^7 | | | 54 | = | 2^66 / 3^38 | | | 79 | = | 2^19 / 3^8 | | |
| 5 | = | 2^4 | / | 3^1 | 30 | = | 2^62 / 3^36 | | | 55 | = | 2^47 / 3^26 | | | 80 | = | 2^84 / 3^49 | | |
| 6 | = | 2^9 | / | 3^4 | 31 | = | 2^24 / 3^12 | | | 56 | = | 2^9 / 3^2 | | | 81 | = | 2^65 / 3^37 | | |
| 7 | = | 2^6 | / | 3^2 | 32 | = | 2^5 / 3^0 | | | 57 | = | 2^74 / 3^43 | | | 82 | = | 2^130 / 3^78 | | |
| 8 | = | 2^3 | / | 3^0 | 33 | = | 2^13 / 3^5 | | | 58 | = | 2^55 / 3^31 | | | 83 | = | 2^46 / 3^25 | | |
| 9 | = | 2^8 | / | 3^3 | 34 | = | 2^59 / 3^34 | | | 59 | = | 2^17 / 3^7 | | | 84 | = | 2^27 / 3^13 | | |
| 10 = | | 2^5 / 3^1 | | | 35 | = | 2^21 | / | 3^10 | 60 | = | 2^82 | / | 3^48 | 85 | = | 2^92 | / | 3^54 |
| 11 = | | 2^13 / 3^6 | | | 36 | = | 2^48 | / | 3^27 | 61 | = | 2^63 | / | 3^36 | 86 | = | 2^73 | / | 3^42 |
| 12 = | | 2^10 / 3^4 | | | 37 | = | 2^10 | / | 3^3 | 62 | = | 2^44 | / | 3^24 | 87 | = | 2^54 | / | 3^30 |
| 13 = | | 2^18 / 3^9 | | | 38 | = | 2^56 | / | 3^32 | 63 | = | 2^25 | / | 3^12 | 88 | = | 2^35 | / | 3^18 |
| 14 = | | 2^7 / 3^2 | | | 39 | = | 2^18 | / | 3^8 | 64 | = | 2^71 | / | 3^41 | 89 | = | 2^16 | / | 3^6 |
| 15 = | | 2^23 / 3^12 | | | 40 | = | 2^64 | / | 3^37 | 65 | = | 2^52 | / | 3^29 | 90 | = | 2^81 | / | 3^47 |
| 16 = | | 2^4 / 3^0 | | | 41 | = | 2^45 | / | 3^25 | 66 | = | 2^33 | / | 3^17 | 91 | = | 2^146 / 3^88 | | |
| 17 = | | 2^20 / 3^10 | | | 42 | = | 2^26 | / | 3^13 | 67 | = | 2^14 | / | 3^5 | 92 | = | 2^62 / 3^35 | | |
| 18 = | | 2^9 / 3^3 | | | 43 | = | 2^53 | / | 3^30 | 68 | = | 2^79 | / | 3^46 | 93 | = | 2^43 / 3^23 | | |
| 19 = | | 2^17 / 3^8 | | | 44 | = | 2^15 | / | 3^6 | 69 | = | 2^60 | / | 3^34 | 94 | = | 2^24 / 3^11 | | |
| 20 = | | 2^44 / 3^25 | | | 45 | = | 2^80 | / | 3^47 | 70 | = | 2^41 | / | 3^22 | 95 | = | 2^89 / 3^52 | | |
| 21 = | | 2^6 / 3^1 | | | 46 | = | 2^42 | / | 3^23 | 71 | = | 2^22 | / | 3^10 | 96 | = | 2^154 / 3^93 | | |
| 22 = | | 2^14 / 3^6 | | | 47 | = | 2^23 | / | 3^11 | 72 | = | 2^68 | / | 3^39 | 97 | = | 2^70 / 3^40 | | |
| 23 = | | 2^22 / 3^11 | | | 48 | = | 2^69 | / | 3^40 | 73 | = | 2^49 | / | 3^27 | 98 | = | 2^51 / 3^28 | | |
| 24 = | | 2^30 / 3^16 | | | 49 | = | 2^31 | / | 3^16 | 74 | = | 2^30 | / | 3^15 | 99 | = | 2^32 / 3^16 | | |
| 25 | = | 2^11 | | / 3^4 | 50 | = | 2^12 | / | 3^4 | 75 | = | 2^95 | / | 3^56 | 100 = 2^97 / 3^57 | | | | |

Após a obtenção destes valores, estes foram testados com ferramentas capazes de lidar com os grandes números envolvidos, e todos os resultados foram confirmados corretos.

*Por favor, não esqueça de apresentar os resultados, afinal deve ser isso que seu leitor quer ver!*

# Conclusões

As abordagens iniciais, mesmo oferecendo apenas resultados parciais, contribuíram bastante para o entendimento do problema e abriram caminho para a solução definitiva. Esta se mostrou bastante simples e eficiente, embora tenha exigido um estudo de como o erro se propaga quando as regras são aplicadas a novos números. A solução implementada não se preocupou em achar sempre os menores expoentes possíveis, mas mesmo assim ela foi bastante eficiente e os expoentes encontrados não são excessivamente grandes (foram feitas comparações com o primeiro algoritmo apresentado, e as diferenças foram poucas). Além disso, a solução recursiva adotada foi razoavelmente elegante e clara, não precisando de uma implementação complicada.

Embora não tenham sido mostrados neste artigo, foram feitos testes com outros valores iniciais (*a* = 1) e para números indo até bem mais do que 100, o algoritmo também funcionou bem.

/

Acreditamos ter desenvolvido uma solução interessante e barata a um problema relativamente complexo, já que em uma linguagem de programação usual não existe suporte direto para tratar com os números envolvidos, e conseguimos determinar resultados corretos que envolvem grandes núme- ros.

*Suas conclusões não precisam ser geniais nem mudar o mundo. Veja que a parte inteligente do artigo já foi feita, e veio antes! Agora é a hora de fechar e dizer o que você acha que poderia vir depois, etc etc. E se você está pensando em dizer coisas como “Provavelmente existem soluções*

*melhores”, então diga ao menos o que você pensa que pode melhorar (por exemplo, neste artigo eu gostaria de evitar o uso de números em ponto flutuante e preferiria trocar por algum tipo de controle com inteiros). Se você não der opções e só disser que dá pra fazer melhor, está se diminundo de graça. Pra que isso?*

*A bibliografia abaixo não foi usada neste artigo, e serve apenas de exemplo para a formatação.*

*Outros exemplos estão na página da biblioteca da PUCRS.*

# Outros detalhes

*(Coisas que não entram num artigo, mas que vou escrever aqui porque deve ser ditas em algum lugar).*

* 1. Em primeiríssimo lugar: você está numa Universidade e portanto deve ter ido à escola por uns onze anos pelo menos. Você teve aulas de português por cerca de sete desses anos, várias vezes por semana e ainda prestou prova para o vestibular. Seu editor de texto, por outro lado, não teve aula nenhuma e provavelmente foi (meio mal) adaptado do inglês. Mostre um pouco de respeito por si mesmo e desligue **tudo o que puder**. Desligue todas as correções automáticas e assuma a responsabilidade pelo que você sabe.
  2. Acentos ainda existem em português, e deverão continuar por muito tempo. Use-os.
  3. Note a consistência no que está escrito: o valor inicial *a*, por exemplo, está sempre escrito da mesma forma, em itálico, para que o leitor não encontre cinco tipos de letra *a* diferentes e tenha de lembrar o que cada uma significa. Isto mostra que o autor teve cuidado ao escrever.
  4. Note que os fontes são sempre coerentes: texto em Times, matemática em itálico, algoritmos em Typewriter. Não há fontes maiores ou menores, nem paragrafos com fonte diferente, margem ou espaçamento maiores ou menores do que os outros.
  5. Veja também que as potências estão sempre escritas corretamente no texto: *xy*. Não se usam notações como *x^y* porque elas são feias. Esta notação só é usada quando se fala do formato de saída, pois ele é exigido desta forma. Da mesma forma, quando são necessários subscritos eles devem ser escritos corretamente: *Ai*, *bi j* etc. Escrever *Aj* ou *bij* é um mau sinal.
  6. Se algo está nas referências deve estar citado no texto, e se está citado no texto tem de aparecer nas referências!! A maneira correta de citar uma referência é: “Segundo [3] e [4], o algoritmo precisa. . . ”
  7. Tenha cuidado com seu texto, ele deve transmitir a impressão de que foi escrito com todo o cuidado do mundo. Dedique tempo a ele e tente deixá-lo melhor, pois sua reputação (e nota) dependem disso.
  8. Este artigo foi totalmente escrito usando LYX, um dos melhores editores de textos que existe, fazendo uso de TEX e LATEX, que são sem dúvida os melhores formatadores de texto do mundo. Tudo isto é software livre rodando sob LINUX. Aprenda a usá-los, vai fazer bem.
  9. Para ajudar na missão ingrata de fazer um trabalho agradável para a leitura, este *checklist* pode ajudar:

Q Artigos não tem capa.

Q Coloquei título neste trabalho? Coloquei meu nome?

Q Usei um espaçamento legal e tamanho de fonte decente ou tem só cinco linhas por página e as letras parecem manchetes?

Q O trabalho tem uma introdução, para que se entenda do que estou falando? Eu contei o que ia resolver?

Q O desenvolvimento é coerente, ou parece que recortei trechos de jornal e de meus colegas e grudei tudo?

Q Dá pra entender minhas explicações ou tem que ter poderes paranormais?

Q Tem o mínimo de código possível? Não dá pra ter ainda menos? Dá pra não ter nenhum? Se não tem nenhum, será que deveria ter um pouco mostrando as partes mais importantes? (Geralmente deveria)

Q Se tem código, eu fico falando durante páginas e páginas entorpecentes e sem fim pra explicar o que ele faz, linha por linha? Posso ser **claro** sem ser **chato**?

Q Se o trabalho pedia a resposta para algum problema, será que esqueci de dar o resultado?

Tinha que dar vários exemplos e testes?

Q Usei figuras pra esclarecer coisas que eram confusas, como aquelas listas que usei e agora eu mesmo mal consigo entender? Como vou querer que outra pessoa entenda?

Q Minhas figuras são claras? Mostram coisas acontecendo passo a passo ou estão jogadas no papel? As legendas são descritivas?

Q Os dados ficam melhor se forem apresentados em tabelas? Tenho gráficos de tempo e desempenho, se preciso? Com legendas, eixos, unidades, ou está uma bagunça feita no Excel?

Q Lembrei de colocar minhas conclusões ao final do trabalho?

Q Coloque as minhas fontes de informação na bibliografia?