MC3305
Algoritmos e Estruturas de Dados II

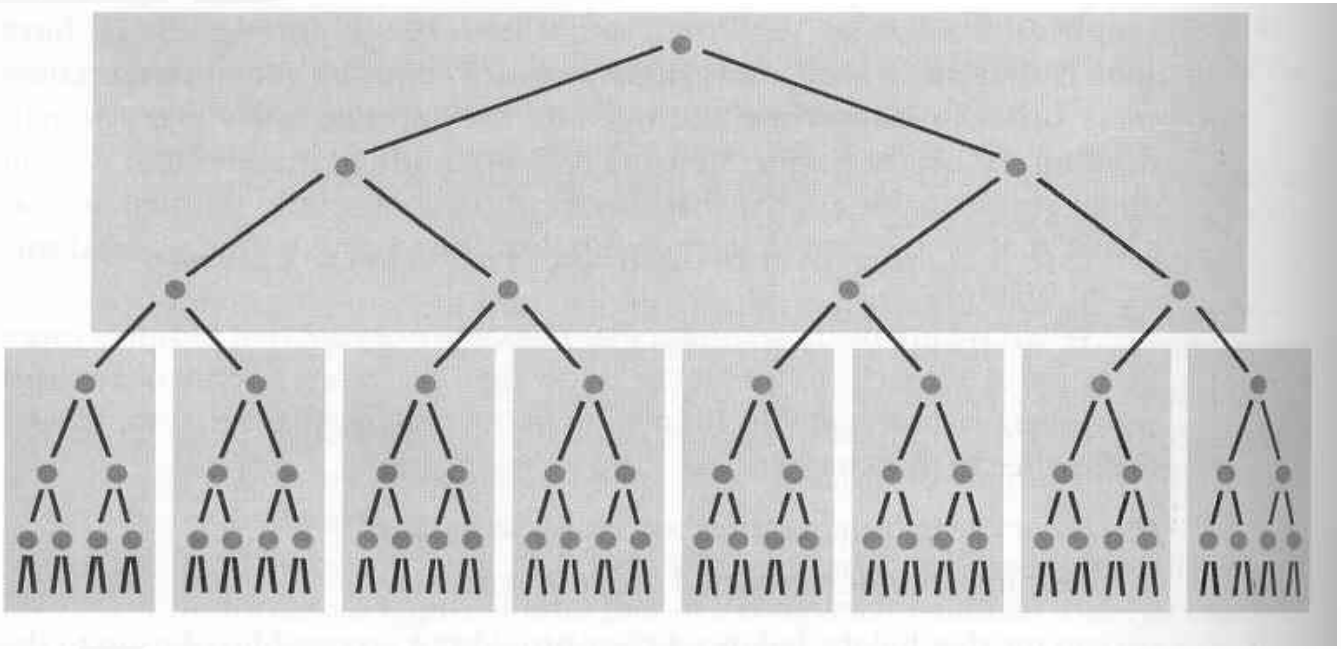
Aula 18 –
Árvores B e parâmetros de compilação

Prof. Jesús P. Mena-Chalco
jesus.mena@ufabc.edu.br

2Q-2015

Árvores binárias paginadas

Exemplo: árvore de 63 nós (página de 7 nós).



Com um nível adicional (3 seeks) é possível acessar 511 nós. Com mais outro nível (4 seeks), é possível acessar 4.095 nós.

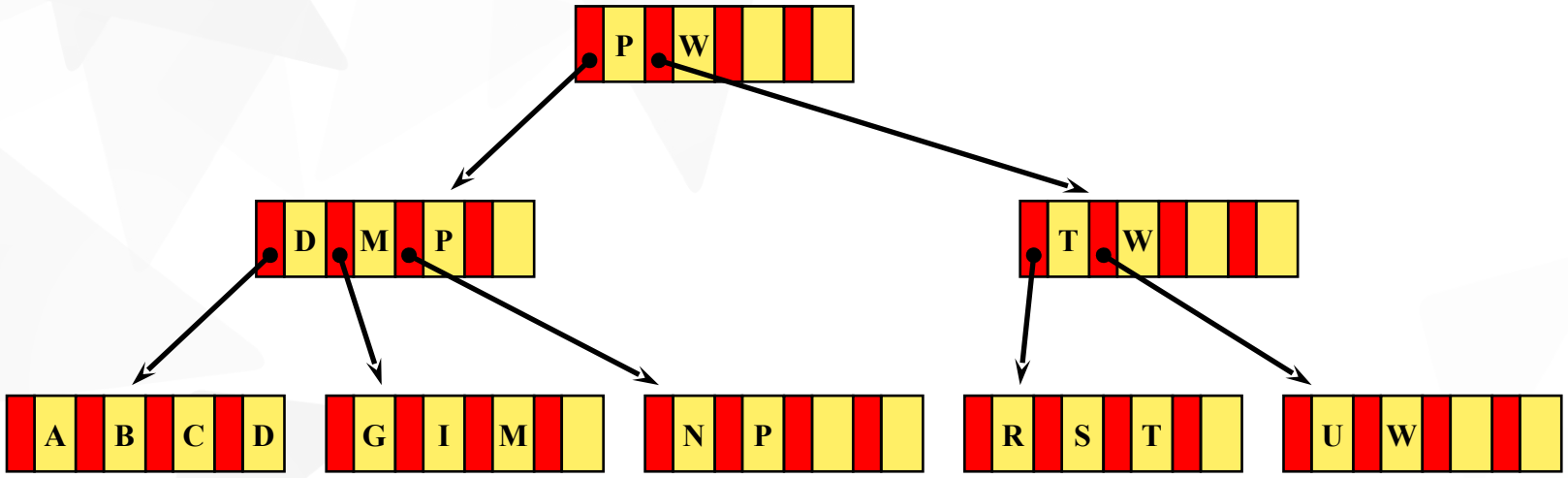
Uma busca binária em 4.095 elementos necessita de **12 acessos**.

Árvores binárias paginadas

O uso do disco pelas árvores paginadas ainda é ineficiente.

- Não há razão para se usar uma árvore dentro de uma página!
  - Grande parte do espaço é desperdiçada em ponteiros.
    - Uma chave, 2 ponteiros
  - Possível busca binária em vetor, em memória: sem acessos ao disco.
    - Lembre-se: o caro é acesso ao disco

Exemplo de árvore B



- Uma árvore B com tamanho de página = 64 ao indexar um arquivo com 1 milhão de registros possibilita achar a chave de qualquer registro em até **3 acessos** ao disco ($\log_{64}1000000$).

- Busca binária = **20 acessos** ($\log_{2}1000000$)

Exemplo

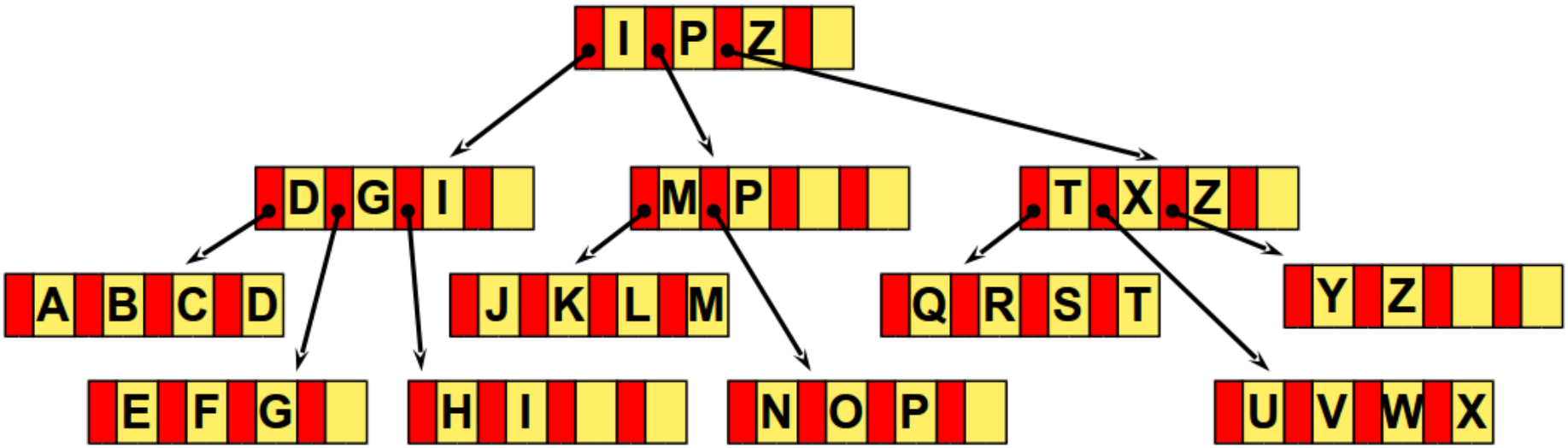
Uso da seguinte sequência:

- **C S D T A M P I B W N G U R K E H O L J Y Q Z F X V.**
- Árvore B de ordem 4 (m=4).
- As 26 chaves são inseridas em uma árvore de altura 3.
  - O número de nós afetados por qualquer inserção nunca é maior do que dois nós por nível (um mudado e outro criado por uma divisão).
  - Com uma ordem maior a árvore é mais “baixa”

Ordem: Número máximo de chaves armazenadas em um nó.

Cada nó deve ter um número mínimo de chaves (exceção a raiz): ~metade da ordem.

Divide o nó cheio em 2 com metade das chaves.



Busca e inserção

Busca

- Busca iterativa, carregando a página na memória, pesquisando na página pela chave e carregando páginas em níveis mais baixos até que se alcance o nível das folhas.
  - Na folha se faz uma busca pela chave procurada.
  - Número máximo de buscas igual à altura da árvore

Inserção

- Duas importantes observações:
  - Começa com uma busca que vai até o nível das folhas.
  - Depois de descobrir o local da inserção no nível das folhas, o trabalho de inserção, detecção de overflow e divisão se propaga para cima.
- Pode ser necessário criar um novo nó raiz
  - caso a raiz atual seja dividida.
- Normalmente ordem é >> 4: divisões relativamente raras
  - Mesmo assim, divisões são O(altura da árvore)

Altura de uma árvore B de ordem m

- Cada página tem no máximo m descendentes.
- Cada página, exceto a raiz e as folhas, tem, no mínimo, $\lceil m/2 \rceil$ descendentes.
- A raiz tem, no mínimo, dois descendentes
  (a não ser que também seja uma folha).

- Todas as folhas aparecem no nível das folhas (altura da árvore).

Altura de uma árvore B de ordem m

Pior caso: quando toda página da árvore tem somente o número mínimo de descendentes:

- Raiz (nível 1) => 2 descendentes.
- Nível 2 => $2\lceil m/2 \rceil$ descendentes.
- Nível 3 => $2\lceil m/2 \rceil^{2}$ descendentes.
- Nível h => $2\lceil m/2 \rceil^{h-1}$ descendentes.
- Para uma árvore de *N* chaves em suas folhas:

  $N \geq 2\lceil m/2 \rceil^{h-1}$.

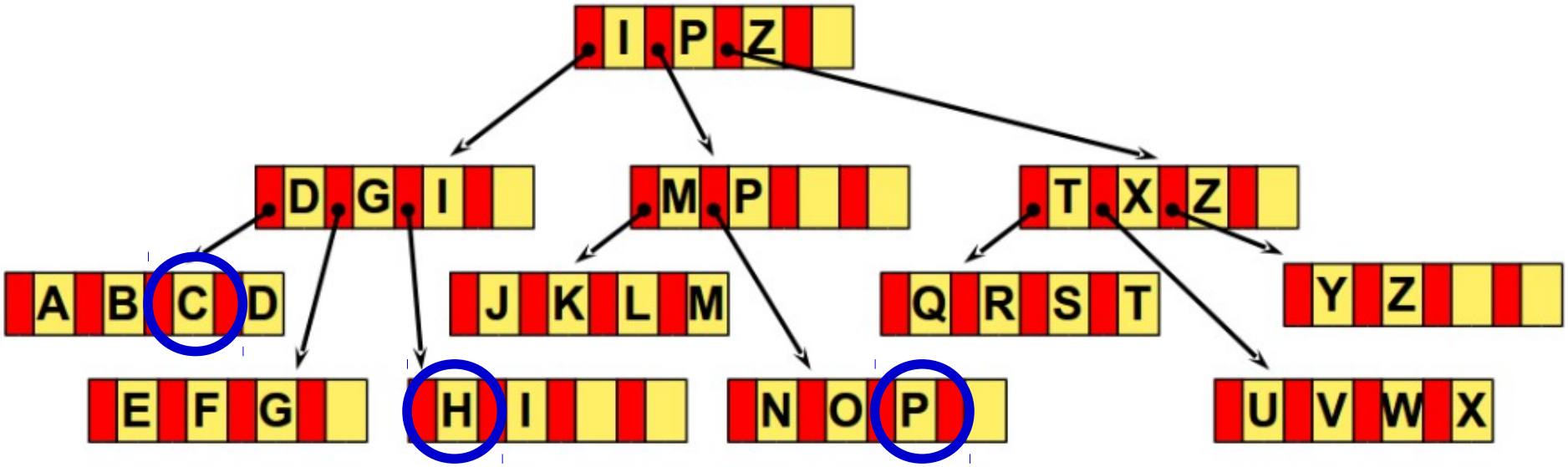
  O que resulta em $h \leq 1 + \log_{\lceil m/2 \rceil} (N/2)$.

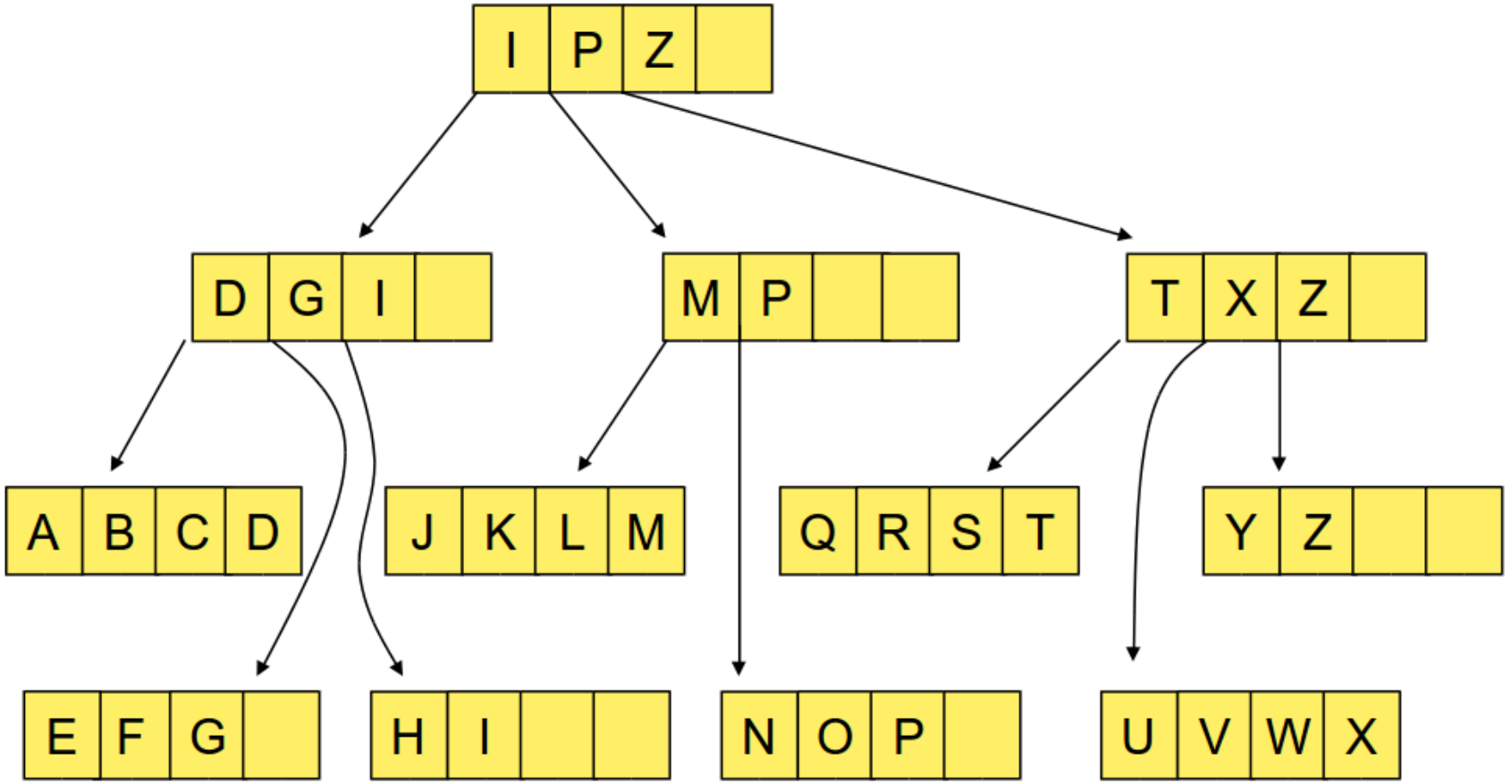
Exemplo:

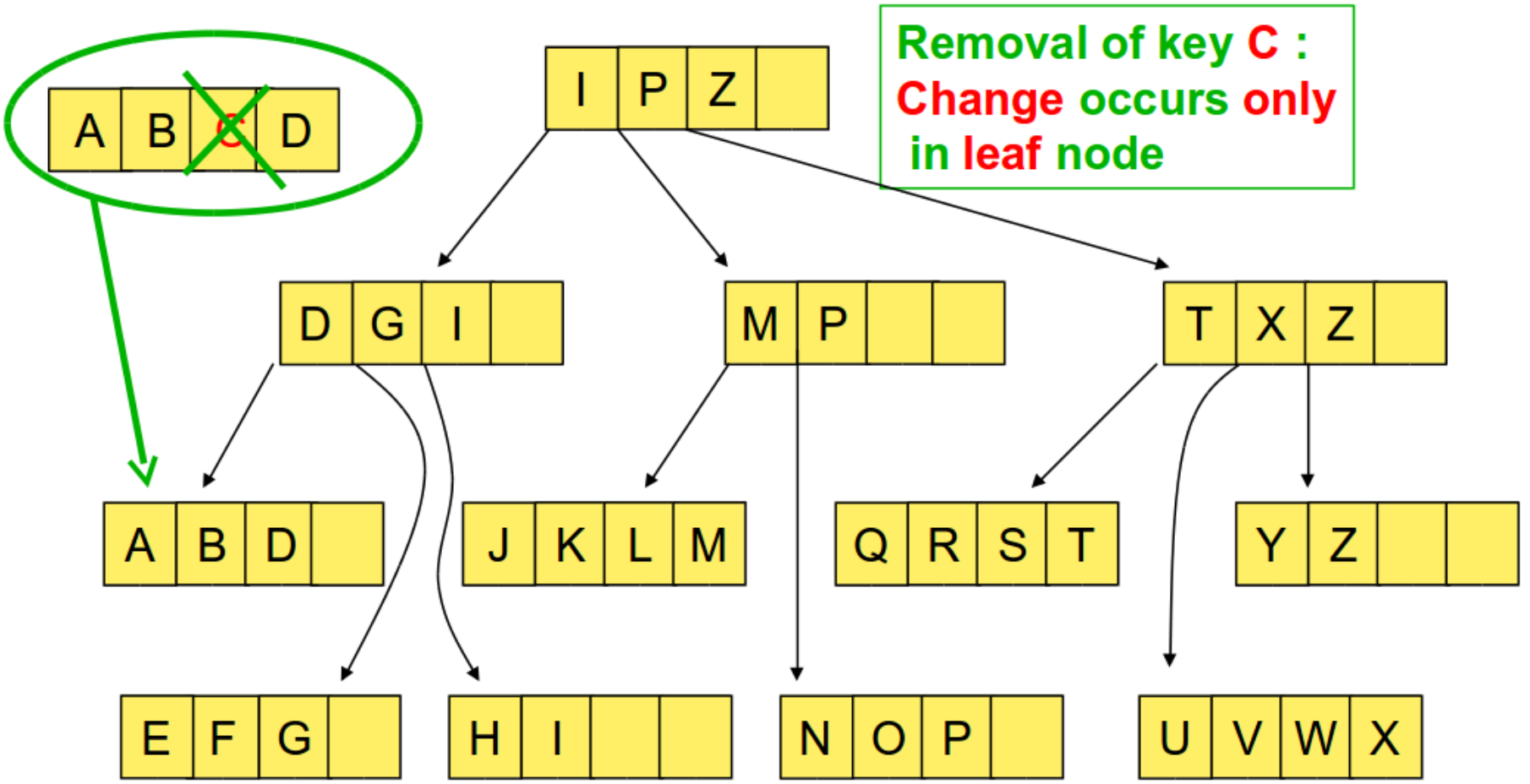
Árvore de ordem 512 que contém 1 milhão de chaves.

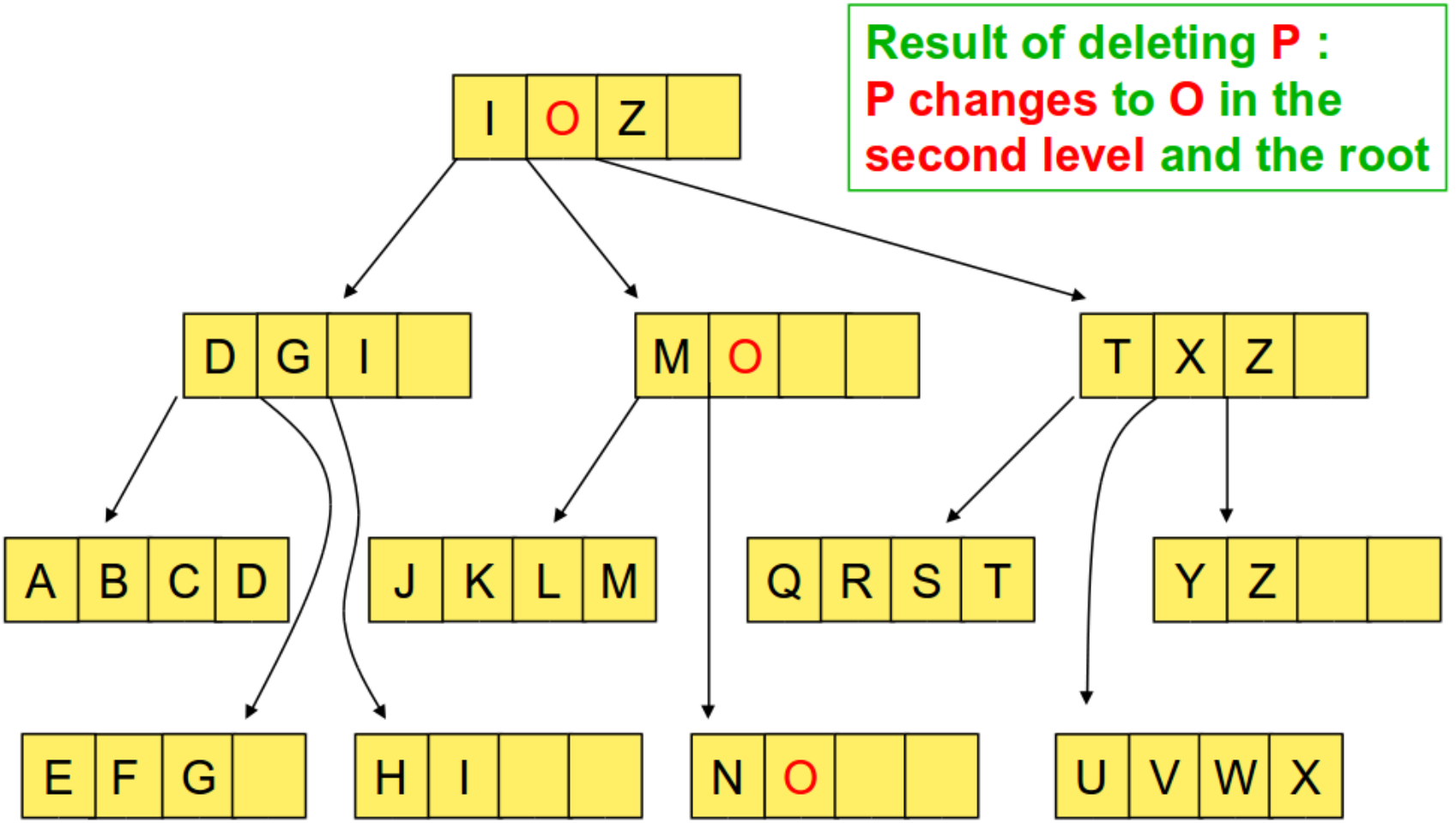
$h \leq \log_{256} 500.000 = 3,37$.

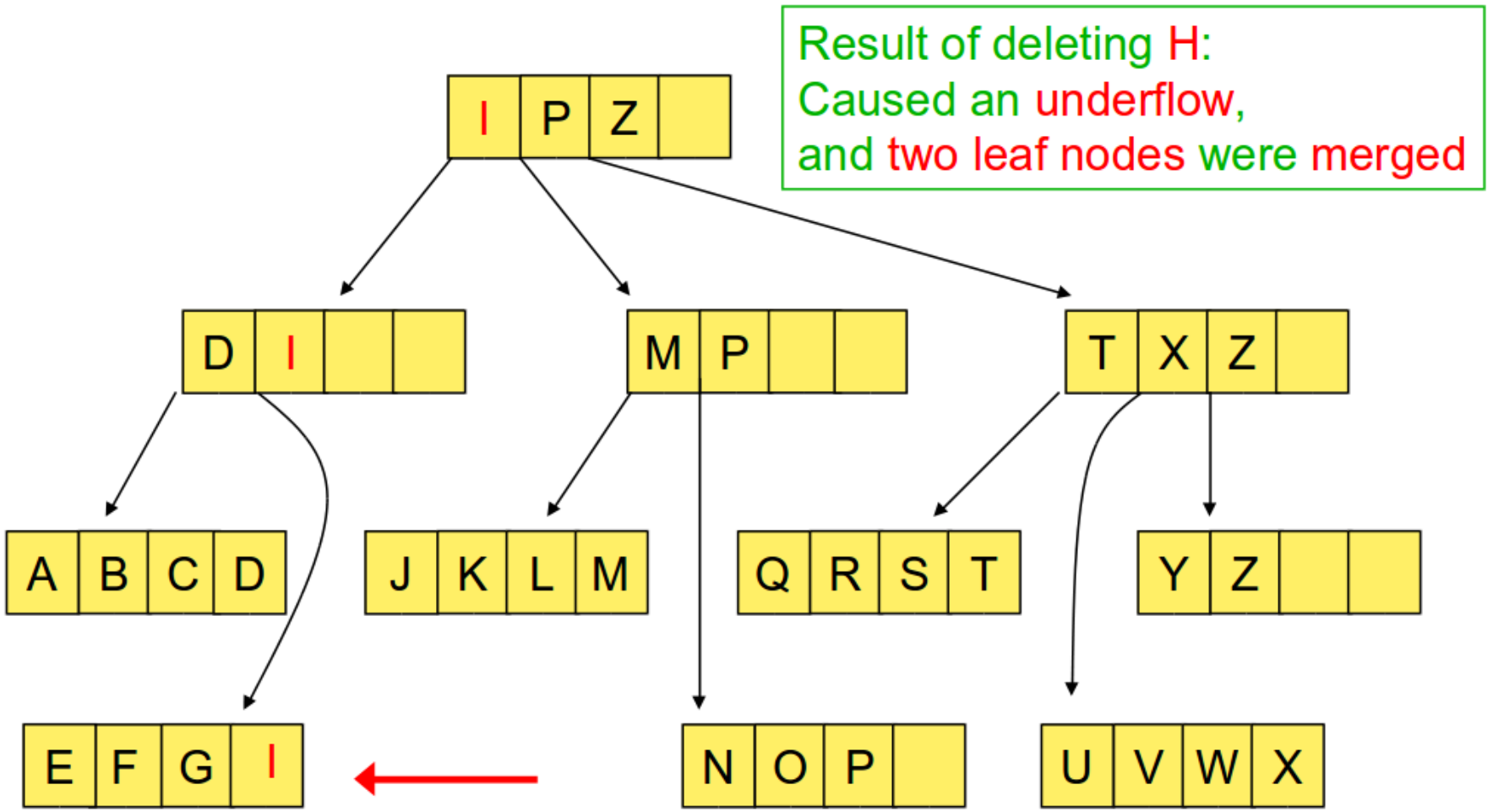
Remoção de chaves




I P Z
D G I
M P
T X Z
A B C D
J K L M
Q R S T
Y Z
E F G
H I
N O P
U V W X


A B C D
I P Z
Removal of key C :
Change occurs only
in leaf node
D G I
M P
T X Z
A B D
J K L M
Q R S T
Y Z
E F G
H I
N O P
U V W X


Result of deleting P :
P changes to O in the
second level and the root
I O Z
D G I
M O
T X Z
A B C D
J K L M
Q R S T
Y Z
E F G
H I
N O
U V W X


Result of deleting H:
Caused an underflow,
and two leaf nodes were merged
I P Z
D I
M P
T X Z
A B C D
J K L M
Q R S T
Y Z
E F G I
N O P
U V W X

Historical Note

- B-tree : Bayer & McCreight
- B^+-tree: Comer
- B*-tree : Knuth, B-trees with 67% minimum occupancy
- $B^{\div}$-trees : B^+-trees with 67% minimum occupancy

Estado-da-arte →

Desafio 03 – opcional – 0,5 na média final da disciplina
Envio até 13/08 (23h50-Tidia)

Atualmente existem muitas estruturas de dados baseadas em árvores que foram propostas para fins computacionais específicos. Qual é o estado-da-arte desse tipo de estruturas? O que está sendo utilizado para grandes projetos (e.g., cloud, e-Science, bigdata)?

Neste desafio o aluno é convidado para estudar uma estrutura de dados em árvores que atualmente esteja sendo utilizada para aplicações. Qual a complexidade computacional para construção da árvore? Qual a complexidade para a busca, inserção e remoção de elementos? A estrutura funciona em memória principal? Quais as vantagens e desvantagens da utilização dessa estrutura?

Deverá ser entregue um relatório técnico que contenha, no mínimo, 2000 palavras.

- Apresentação livre (quanto mais completo melhor).
- Apresente um resumo e contextualização histórica. Referencie todos os trabalhos.
- Apresentação individual.
- Os melhores trabalhos serão publicados na página da disciplina.

Algumas considerações para melhora do desempenho → otimização

tempo.c

```c
#include<stdio.h>
#include<time.h>

int main()
{
    int i, k, n=100;
    double acc=0;
    clock_t t1, t2;

    for (k=0; k<n; k++) {
        t1 = clock();
        srand(time(NULL));
        for (i=0; i<1000000; i++) {
            rand();
        }
        t2 = clock();

        printf("%.6f\n", (double)(t2-t1)/CLOCKS_PER_SEC);
        acc += (double)(t2-t1)/CLOCKS_PER_SEC;
    }

    printf("Tempo medio: %f\n", acc/n);
}
```

tempo.c

```
$ gcc tempo.c -o tempo.exe
$ ./tempo.exe
…
0.009594
0.009857
0.010107
0.009947
0.009868
0.009669
0.010100
0.010710
0.010156
0.010282
0.012067

Tempo medio: 0.012217
```

tempo.c

```
$ gcc tempo.c -o tempo.exe
$ ./tempo.exe
…
0.009594
0.009857
0.010107
0.009947
0.009868
0.009669
0.010100
0.010710
0.010156
0.010282
0.012067

Tempo medio: 0.012217
```

```
$ gcc tempo.c -o tempo.exe -Ofast
$ ./tempo.exe
…
0.013685
0.013098
0.012659
0.012056
0.010953
0.012111
0.013188
0.010750
0.012067
0.012273
0.012887

Tempo medio: 0.010800
```

tempo.c

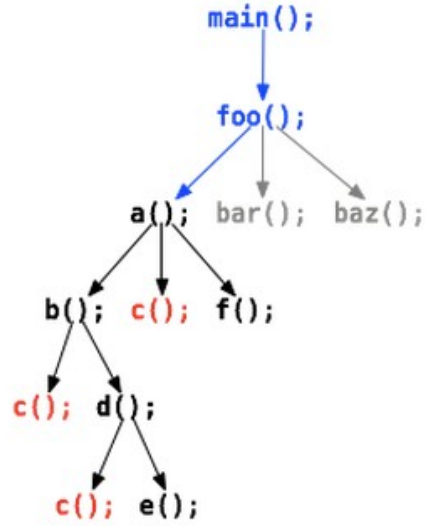
```
$ time ./tempo.exe
…
0.012891
0.012949
0.013130
0.012945
Tempo medio: 0.010784

real    0m1.093s
user    0m1.068s
sys     0m0.012s
```

Otimização

```
-O0
-O1
-O2
-O3
-Ofast
-Os   ← Optimize for size
```

```
void a() {
   b();
   c();
   f();
}

void b() {
   c();
   d();
}

void c() {
   //...
}
```

```
void d() {
   c();
   e();
}

void e() {
   //...
}

void f() {
   //...
}

//...
```



tempo.c

```
$ gcc tempo.c -o tempo1.exe
$ gcc tempo.c -o tempo2.exe -Ofast
$ gcc tempo.c -o tempo3.exe -Os

$ ls -l
-rwxr-xr-x 1 ubuntu ubuntu  8715 Jul 28 09:36 tempo1.exe*
-rwxr-xr-x 1 ubuntu ubuntu 10205 Jul 28 09:36 tempo2.exe*
-rwxr-xr-x 1 ubuntu ubuntu  8721 Jul 28 09:36 tempo3.exe*
```

fibo.c

```c
#include<stdio.h>
#include<time.h>

int f(int n) {
    if (n<=1)
        return n;
    else
        return f(n-1)+f(n-2);
}

int main()
{
    clock_t t1, t2;

    t1 = clock();
    printf("%d\n", f(42) );
    t2 = clock();

    printf("Tempo: %f\n", (double)(t2-t1)/CLOCKS_PER_SEC);
}
```

```
gcc fibo.c -ofibo.exe

$ ./fibo.exe
267914296
Tempo: 2.735228
```

```
gcc fibo.c -ofibo.exe -Ofast

$ ./fibo.exe
267914296
Tempo: 1.825731
```

```
$ gcc -Wall -pg  fibo.c  -ofibo.exe

$ ./fibo.exe

$ gprof fibo.exe > fibo.txt
```

Wall ← Warnings all
pg ← Para uso com o gprof

Gera um arquivo gmoun.out

Gera um arquivo fibo.txt

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 94.72      2.28     2.28        1     2.28     2.28  f
  4.62      2.39     0.11                             frame_dummy

 %         the percentage of the total running time of the
time       program used by this function.
```

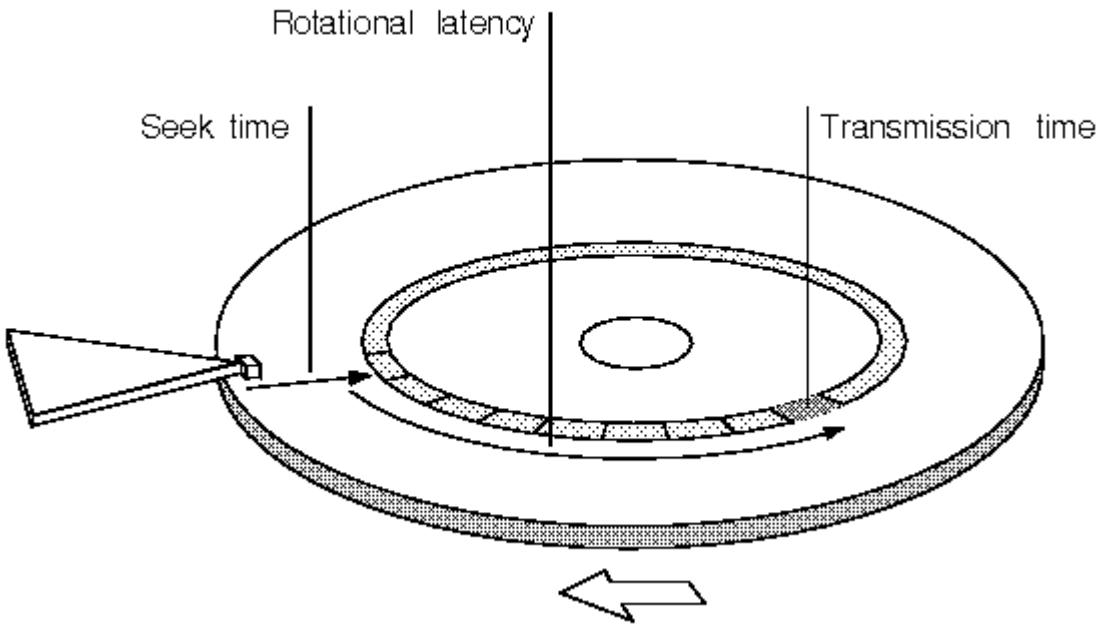
```
index % time    self  children    called     name
                                866988872             f [1]
                2.28    0.00       1/1           main [2]
[1]     95.3    2.28    0.00       1+866988872 f [1]
                                866988872             f [1]
-----------------------------------------------
                                                 <spontaneous>
[2]     95.3    0.00    2.28                  main [2]
                2.28    0.00       1/1           f [1]
-----------------------------------------------
                                                 <spontaneous>
[3]      4.7    0.11    0.00                  frame_dummy [3]
-----------------------------------------------
```

Custo de acesso a disco

Custo do acesso a disco

O custo do acesso a disco pode ser dividido em três operações distintas

- Tempo de busca (seek).
- Atraso rotacional no disco.
- Tempo de transferência dos dados.



Comparação memória RAM vs discos

- **Discos são lentos**
  - Um acesso na memória RAM (Random Access Memory) tipicamente pode ser feito em 30 a 60 nanosegundos (30 x10^{-9} s).
  - Obter a mesma informação no disco tipicamente leva de 7 a 10 milisegundos (7 x10-3 s).
  - Memória RAM é cerca de 1 milhão de vezes mais rápida que o disco.
- Mas provêm uma capacidade maior a um preço menor.
- Também são capazes de reter dados quando o computador é desligado.

```
$ mount
/dev/sda1 on / type ext4 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
none on /sys/fs/cgroup type tmpfs (rw)
none on /sys/fs/fuse/connections type fusectl (rw)
none on /sys/kernel/debug type debugfs (rw)
none on /sys/kernel/security type securityfs (rw)
udev on /dev type devtmpfs (rw,mode=0755)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
tmpfs on /run type tmpfs (rw,noexec,nosuid,size=10%,mode=0755)
none on /run/lock type tmpfs (rw,noexec,nosuid,nodev,size=5242880)
none on /run/shm type tmpfs (rw,nosuid,nodev)
none on /run/user type tmpfs (rw,noexec,nosuid,nodev,size=104857600,mode=0755)
none on /sys/fs/pstore type pstore (rw)
/dev/sda3 on /backups type ext4 (rw)
systemd on /sys/fs/cgroup/systemd type cgroup (rw,noexec,nosuid,nodev,none,name=systemd)
```

```
$ sudo hdparm -I /dev/sda1

/dev/sda1:

ATA device, with non-removable media
        Model Number:       ST31000524AS
        Serial Number:      6VPKSXYB
        Firmware Revision:  JC4A
        Transport:          Serial, SATA Rev 3.0
Standards:
        Used: unknown (minor revision code 0x0029)
        Supported: 8 7 6 5
        Likely used: 8
Configuration:
        Logical         max     current
        cylinders       16383   16383
        heads           16      16
        sectors/track   63      63
        --
        CHS current addressable sectors:   16514064
        LBA    user addressable sectors:  268435455
        LBA48  user addressable sectors: 1953525168
        Logical/Physical Sector size:           512 bytes
        device size with M = 1024*1024:      953869 MBytes
        device size with M = 1000*1000:     1000204 MBytes (1000 GB)
        cache/buffer size  = unknown
        Nominal Media Rotation Rate: 7200
Capabilities:
        LBA, IORDY(can be disabled)
        Queue depth: 32
        Standby timer values: spec'd by Standard, no device specific minimum
        R/W multiple sector transfer: Max = 16  Current = 16
        Recommended acoustic management value: 208, current value: 208
        DMA: mdma0 mdma1 mdma2 udma0 udma1 udma2 udma3 udma4 udma5 *udma6
             Cycle time: min=120ns recommended=120ns
        PIO: pio0 pio1 pio2 pio3 pio4
             Cycle time: no flow control=120ns  IORDY flow control=120ns
```

Desafio 03 – opcional – 0,5 na média final da disciplina
Envio até 13/08 (23h50-Tidia)

Atualmente existem muitas estruturas de dados baseadas em árvores que foram propostas para fins computacionais específicos. Qual é o estado-da-arte desse tipo de estruturas? O que está sendo utilizado para grandes projetos (e.g., cloud, e-Science, bigdata)?

Neste desafio o aluno é convidado para estudar uma estrutura de dados em árvores que atualmente esteja sendo utilizada para aplicações. Qual a complexidade computacional para construção da árvore? Qual a complexidade para a busca, inserção e remoção de elementos? A estrutura funciona em memória principal? Quais as vantagens e desvantagens da utilização dessa estrutura?

Deverá ser entregue um relatório técnico que contenha, no mínimo, 2000 palavras.

- Apresentação livre (quanto mais completo melhor).
- Apresente um resumo e contextualização histórica. Referencie todos os trabalhos.
- Apresentação individual.
- Os melhores trabalhos serão publicados na página da disciplina.