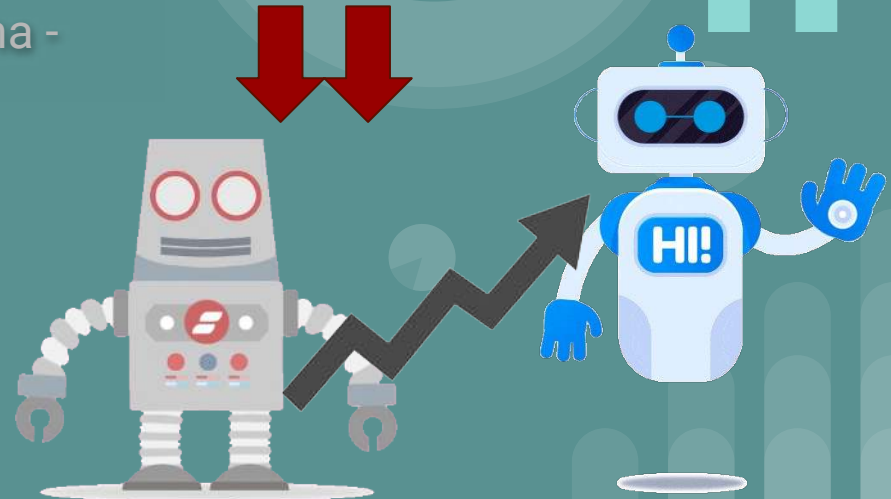


[T3] Análise e Otimização - Desenvolvimento de SW embarcado -

Efetuar medições de parâmetros (tempo e memória) de uma aplicação embarcada em diferentes plataformas e otimizar um destes na mesma. -

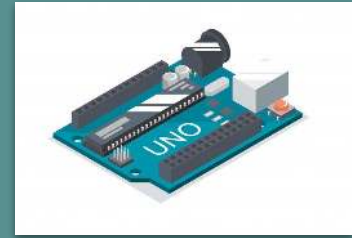
Equipe: João Gabriel & Alyson Noronha -

```
( ) [ ] [ ]  
[ ] [ ]  
[ ] [ ] [ ]  
[ ] [ ]
```



[Otimização Proposta] - Considerações Iniciais -

- Plataforma escolhida para otimização -
 - Arduino Uno.



- [Tempo x Memória] - Parâmetro escolhido para otimização -
 - Memória de código(Flash).
 - { Memória de dados(SRAM) } ← Principalmente! ✓
 - Tempo de computação. } ✗

*
*Muita SRAM
sendo usada!

() [] []
[] []
[] [] []
[] []

codigo_Hungaro_NAO_OTIMIZADO

compilação terminada.

ando a biblioteca SoftwareSerial na versão 1.0 na pasta: Program F:
:\Program Files (x86)\Arduino\hardware\tools\avr-size" -2
sketch usa 8310 bytes (25%) de espaço de armazen para programas
riáveis globais usam 1250 bytes (61%) de memória dinâmica, deixando 7

[Desafios Técnicos] - Medições de Código - Arduino Uno -

- Memórias de Código/ Dados - Como/Onde medir?

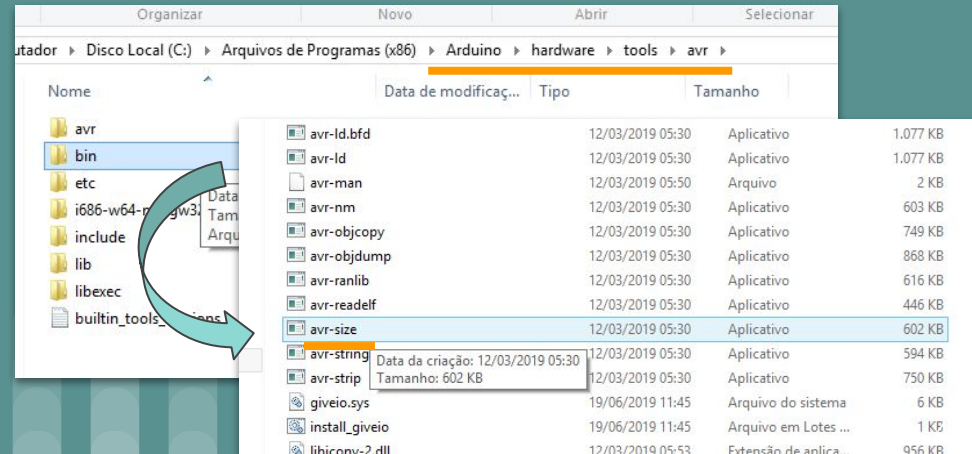
- IDE do Arduino -

- A própria janela da IDE mostra em tempo de compilação diversos dados no console em relação ao arquivo de programa ".ino" inclusive a % de espaço de armazenamento para programas (Flash) e de memória dinâmica (ou de dados - SRAM) ocupados e o total de ocupação permitido para ambos.



- Compilador AVR-GCC -

- Está no diretório da IDE do arduino e contém muitas ferramentas na forma de arquivos executáveis(.exe) para analisar os códigos da plataforma.
- Dispõe da ferramenta 'avr-size' que mostra em sua saída medições de dados e memórias usadas em um programa compilado na IDE como as memórias de código e dados deste.



[Desafios Técnicos] - Medições de Código - Arduino Uno -

- Memórias de Código/ Dados - Observação dos resultados?

*Opção escolhida pela equipe:

- IDE do Arduino -

{ - Compilador AVR-GCC - }



- No próprio console da janela da IDE após terminar uma compilação:

- Após executar o arquivo “avr-size.exe” na janela gerada por este (Que pede como entradas o caminho para onde os arquivos resultantes da compilação na IDE são colocados e o nome do arquivo “.elf” gerado):

```
#define MAXM 45
<

Compilação terminada.

Usando a biblioteca SoftwareSerial na versão 1.0 na pasta: C:\Program Files
"C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avr-size" -A "C:\\Users\\PAULOM-1\\AppData\\
O sketch usa 8310 bytes (25%) de espaço de armazenamento para programas. O máximo são 32256 bytes.
Variáveis globais usam 1250 bytes (61%) de memória dinâmica, deixando 798 bytes para variáveis locais.
<
8
```

```
C:\WINDOWS\System32\cmd.exe
Please enter the path where your arduino is installed: C:\Program Files (x86)\Arduino
Please enter name of your .elf file: codigo_Hungaro_NAO_OTIMIZADO.ino.elf
AVR Memory Usage
-----
Device: atmega328p
Program: 8310 bytes (25.4% Full)
(.text + .data + .bootloader)
Data: 1250 bytes (61.0% Full)
(.data + .bss + .noinit)
Press Enter to close this window
```

← Uso de memória do código no ".elf" no processador "atmega328p" calculado pelo compilador AVR-GCC

← Microcontrolador aonde se faz a medição (aqui é do Arduino Uno)

Memórias de código e dados retornadas

[Dados minimamente mais precisos.]



[Desafios Técnicos] - Medições de Código - Arduino Uno -

- Tempo de computação - Como/Onde medir?

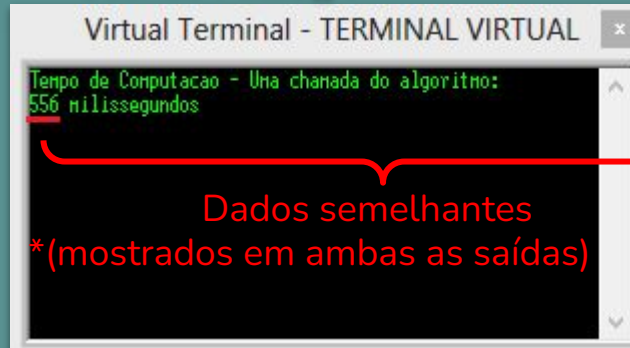
- No próprio código usando-se de funções específicas para os tipos de medições solicitadas.

***Opção escolhida pela equipe:**

- Micros() vs. Millis() -

- Ótimas ferramentas para medir o tempo passado desde o começo da execução do algoritmo pela placa microcontroladora. Diferem quanto a precisão de suas saídas!

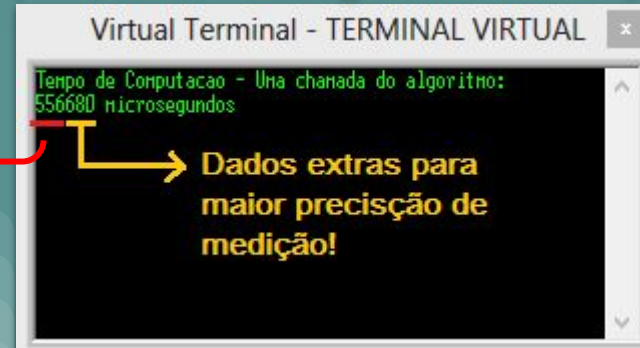
- Saída do tempo de computação - Usando **Millis()** :



```
Virtual Terminal - TERMINAL VIRTUAL x
Tempo de Computacao - Uma chamada do algoritmo:
556 milissegundos
```

Dados semelhantes
*(mostrados em ambas as saídas)

- Saída do tempo de computação - Usando **Micros()** :



```
Virtual Terminal - TERMINAL VIRTUAL x
Tempo de Computacao - Uma chamada do algoritmo:
556680 microsegundos
```

Dados extras para maior precisão de medição!



[Desafios Técnicos] - Medições de Código - Arduino Uno -

- Tempo de computação - "Micros()" - Para uma chamada do algoritmo -

- Esquema de código para medição - Esboço -

- Obs: O dado retornado pela função "Micros()" é do tipo "unsigned long", logo todas as variáveis que a receberem devem ser do mesmo tipo. << (Atentar a conflitos com tipos das variáveis!)

- Abaixo o esqueleto do código para medição do tempo de computação em uma só chamada do algoritmo implementado:

```
{ unsigned long startMicros = micros(); } Variável que recebe o tempo de início (em micros).  
// Espaço reservado para função a ser medida!  
{ unsigned long endMicros = micros(); } Variável que recebe o tempo de término (em micros).  
mySerial.println(F("Tempo de Computação - Uma chamada do algoritmo: "));  
( ) [ ] { mySerial.print(endMicros-startMicros); } Subtraímos do tempo de término o tempo de início para  
[ ] [ ] mySerial.println(F(" microsegundos")); } chegar no tempo de computação da função sendo medida.  
[ ] [ ]  
[ ] [ ]
```

[Desafios Técnicos] - Medições de Código - Arduino Uno -

- Tempo de computação - “Micros()” - Para uma chamada do algoritmo -

- Esquema de código para medição - No algoritmo implementado - Real -

- Porção do código implementado pela equipe sem o código para medição do tempo de computação:

```
codigo_Hungaro_OTIMIZACAO_01 $
}

void setup() {
  mySerial.begin(9600);
  Serial.begin(9600);
}

void loop() {
  long custo_maximo = 0;

  hungarian();
  for(int j = 1; j <= m; j++) {
    if (pairV[j] == 0) continue;
    if ((j < m) or (pairV[j] < m)) custo_maximo += (int)pgm_read_word_near(&cost[pairV[j]][j]);
  }
  custo_maximo = -custo_maximo;

  mySerial.println(F("Custo maximo da matrix solicitada: "));
  mySerial.print(custo_maximo);
  mySerial.println(F("\n"));

  while(true);
}
```

Trecho que se deseja medir o tempo de computação

- Porção do código implementado pela equipe com o código para medição do tempo de computação:

```
codigo_Hungaro_NAO_OTIMIZADO $
unsigned long timeMillis, timeMillis2, timeMillis3;
void loop() {
  timeMillis = micros();

  long custo_maximo = 0;
  hungarian();
  for(int j = 1; j <= m; j++) {
    if (pairV[j] == 0) continue;
    if ((j < m) or (pairV[j] < m)) custo_maximo += (int)pgm_read_word_near(&cost[pairV[j]][j]);
  }
  custo_maximo = -custo_maximo;

  timeMillis2 = micros();
  timeMillis3 = timeMillis2 - timeMillis;

  mySerial.println(F("Tempo de Computacao - Uma chamada do algoritmo: "));
  mySerial.print(timeMillis3);
  mySerial.println(F(" microsegundos"));
  while(true);
}
```


[Desafios Técnicos] - Medições de Código - Arduino Uno -

- Tempo de computação - “Micros()” - Para 1000 chamadas do algoritmo -
- *Medição adicional << Escolha da equipe

- Esquema de código para medição - Esboço -

- Abaixo o esqueleto do código para medição do tempo de computação em 1000 chamadas do algoritmo implementado:

```
{ unsigned long startMicros = micros(); } Variável que recebe o tempo de início (em micros).  
{  
  for (int n = 0; n < 1000; n++) {  
    // Espaço reservado para função a ser medida!  
  }  
} Variável que recebe o tempo de término (em micros).  
{ unsigned long endMicros = micros(); }  
{ Serial.println((endMicros - startMicros)/1000); }
```

Subtraímos do tempo de término o tempo de início e depois dividimos por 1000 (pois são 1000 repetições) para chegar ao um tempo de computação mais preciso da função sendo medida.

[Desafios Técnicos] - Medições de Código - Arduino Uno -

- Tempo de computação - “Micros()” - Para 1000 chamadas do algoritmo -

- Esquema de código para medição - No algoritmo implementado - Real -

- Porção do código implementado pela equipe sem o código para medição do tempo de computação:

```
codigo_Hungaro_OTIMIZACAO_01 $
}

void setup() {
  mySerial.begin(9600);
  Serial.begin(9600);
}

void loop() {
  long custo_maximo = 0;

  hungarian();
  for(int j = 1; j <= m; j++) {
    if (pairV[j] == 0) continue;
    if ((j < m) || (pairV[j] < m)) custo_maximo += (int)pgm_read_word_near(&cost[pairV[j]][j]);
  }
  custo_maximo = -custo_maximo;

  mySerial.println(F("Custo maximo da matrix solicitada: "));
  mySerial.print(custo_maximo);
  mySerial.println(F("\n"));

  while(true);
}
```

Trecho que se deseja medir o tempo de computação

- Porção do código implementado pela equipe com o código para medição do tempo de computação:

```
codigo_Hungaro_NAO_OTIMIZADO $
}

unsigned long startMicros, endMicros, result;
void loop() {
  startMicros = micros();

  for (int i = 0; i < 1000; i++) {
    long custo_maximo = 0;
    hungarian();
    for(int j = 1; j <= m; j++) {
      if (pairV[j] == 0) continue;
      if ((j < m) || (pairV[j] < m)) custo_maximo += (int)pgm_read_word_near(&cost[pairV[j]][j]);
    }
    custo_maximo = -custo_maximo;
  }

  endMicros = micros();
  result = (endMicros - startMicros)/1000;

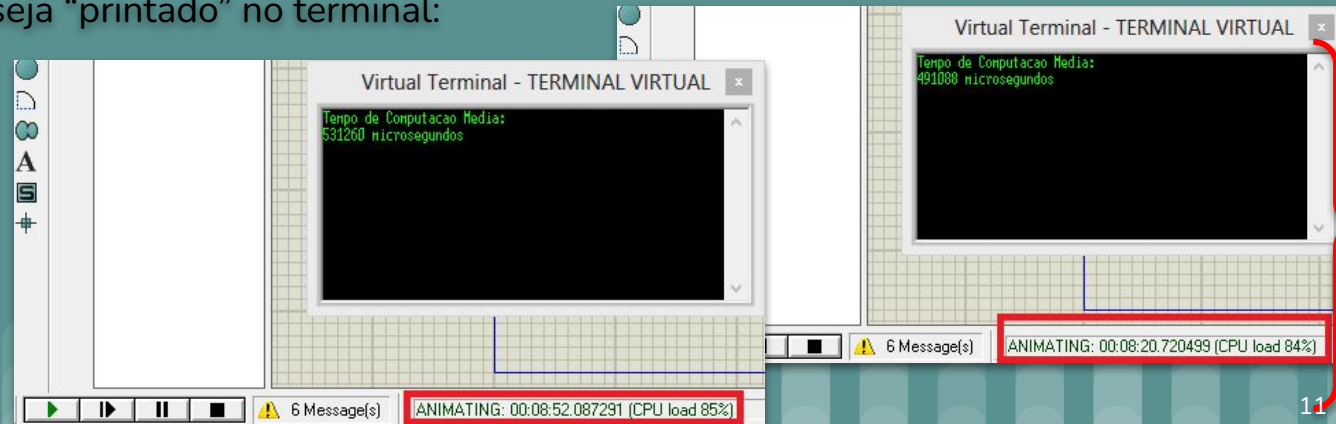
  mySerial.println(F("Tempo de Computacao Media: "));
  mySerial.print(result);
  mySerial.println(F(" microssegundos"));
  while(true);
}
```

[Desafios Técnicos] - Medições de Código - Arduino Uno -

- Tempo de computação - Observação dos resultados?

- Ambiente de simulação - “Proteus Design Suite”(Proteus ISIS) -

- Modo de simulação: IDE Arduino (Providência o código num “.hex”) + Proteus ISIS (Providência o circuito).
- Para a captura do tempo de computação médio precisa-se esperar períodos consideráveis de tempo de simulação no Proteus até que o resultado seja “printado” no terminal:



[Desafios Técnicos] - Medições de Código - Desktop Linux -

- Memórias de Código/ Dados - Como/Onde medir?

- Ambiente de medição:

- Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.81 GHz

- Comando size do *Linux* para medir memória de arquivos:

*Com os dados de entrada (Matriz de entrada de ~2K elementos):

```
davi@davi-System-Product-Name:~/Área de Trabalho/Alysson$ size ./1
{ text    data } bss    dec    hex filename
{ 2630    8724 } 1048   12402  3072 ./1
```

*Sem os dados de entrada (Matriz de entrada de ~2K elementos):

```
davi@davi-System-Product-Name:~/Área de Trabalho/Alysson$ size ./1
{ text    data } bss    dec    hex filename
{ 2630     608 } 9176   12414  307e ./1
```

```
( ) [ ] [ ]
[ ] [ ]
[ ] [ ] [ ]
[ ] [ ]
```

[Desafios Técnicos] - Medições de Código - Desktop Linux -

- Tempo de computação - Como/Onde medir? - (Medido no mesmo ambiente de medição informado do slide passado) -

- Função `clock()` + `macro CLOCKS_PER_SEC` (biblioteca - "`time.h`") do C para medir o tempo de computação:

*Para 1000 iterações do algoritmo:

```
clock_t t = clock();

for(int i=0; i<1000; i++) {
    int custo_maximo = 0;
    hungarian();
    // uso o resultado do Método Húngaro para calcular o custo máximo
    for(int j = 1; j <= m; j++) {
        if (pairV[j] == 0) continue;
        custo_maximo += cost[pairV[j]][j];
    }
    // exibe o oposto da resposta para exibir o resultado correto
    // printf("Custo maximo de: %d\n", -custo_maximo);
}

t = clock() - t;

printf("Tempo: %lf segundos\n", (double) t/CLOCKS_PER_SEC);
```

Tempo: 0.287000 segundos

***Mais preciso** para
1000 iterações!

***Pouca
precisão** !

*Para 1 iteração do algoritmo:

```
clock_t t = clock();

int custo_maximo = 0;
hungarian();
// uso o resultado do Método Húngaro para calcular o custo máximo
for(int j = 1; j <= m; j++) {
    if (pairV[j] == 0) continue;
    custo_maximo += cost[pairV[j]][j];
}

// exibe o oposto da resposta para exibir o resultado correto
// printf("Custo maximo de: %d\n", -custo_maximo);

t = clock() - t;

printf("Tempo: %lf segundos\n", (double) t/CLOCKS_PER_SEC);
```

Tempo: 0.000000 segundos

[Otimização Proposta] - 1ª Implementação -

- Explicação -

- Mudar o tipo de algumas variáveis de "long"(4-bytes) para "int"(2-bytes) tanto globalmente como localmente (devido ao espaço desnecessário de memória SRAM que isso reserva).

- Códigos - Comparativo -

- Sem Implementação -

```
long pu[MAXN], pv[MAXN];
long way[MAXN], minv[MAXN];
bool used[MAXN];

long pairV[MAXN];

void hungarian() {
    memset(&pairV, 0, sizeof pairV);
    for(int i = 1, j0 = 0; i <= n; i++) {
        pairV[0] = i;
        memset(&minv, INF, sizeof minv);
        memset(&used, false, sizeof used);

        do {
            used[j0] = true;
            long i0 = pairV[j0], delta = INF, j1;
            for(int j = 1; j <= m; j++) {
                do {
                    long j1 = way[j0];
                    pairV[j0] = pairV[j1];
                    j0 = j1;
                } while(j0);
            }
            for(int i = 1; i < n; i++) {
                long aux = pairV[i + 1];
                pairV[i + 1] = pairV[i];
                pairV[i] = aux;
            }
        }
    }
}
```



- Depois da 1ª Implementação -

```
long pu[MAXN], pv[MAXN];
long minv[MAXN];
bool used[MAXN];

int pairV[MAXN], way[MAXN]; // <- Modificado! ("long" -> "int").

void hungarian() {
    memset(&pairV, 0, sizeof pairV);
    for(int i = 1, j0 = 0; i <= n; i++) {
        pairV[0] = i;
        memset(&minv, INF, sizeof minv);
        memset(&used, false, sizeof used);

        do {
            used[j0] = true;
            int i0 = pairV[j0], j1; // <- Modificado! ("long" -> "int").
            long delta = INF;

            do {
                int j1 = way[j0]; // <- Modificado! ("long" -> "int").
                pairV[j0] = pairV[j1];
                j0 = j1;
            } while(j0);

            for(int i = 1; i < n; i++) {
                int aux = pairV[i + 1]; // <- Modificado! ("long" -> "int").
                pairV[i + 1] = pairV[i];
                pairV[i] = aux;
            }
        }
    }
}
```

[Otimização Proposta] - 1ª Implementação - Resultados e Medições -

- Ganhos e Perdas (%) -

- Redução de 10,92% no tempo de computação p/ 1 chamada e de 13,53% p/1000 chamadas.
- Redução de 14,40% na memória de dados e de 2,19% na memória de código.

Código	Memória de Código < em bytes >	Memória de Dados < em bytes >	Tempo de Computação (p/ 1 chamada do alg) < em microsegundos (μ) >	Tempo de Computação Médio (p/ 1000 chamadas do alg) < em microsegundos (μ) >
< Arduino Uno > Sem Implementação	8310	1250	556680 (\sim 0,56 segundos)	567906 (\sim 0,57 segundos)
< Arduino Uno > Após a 1ª Implementação	8128	1070	495896 (\sim 0,5 segundos)	491088 (\sim 0,49 segundos)

() [] []
[] []
[] [] []
[] []

- “%” de Ganhos e Perdas calculadas pela ferramenta online:
<https://www.calcul.net/financeiro/calculadora-de-porcentagem-calculo-online/>

[Otimização Proposta] - 2ª Implementação - Explicação -

- Explicação -

- Mudar o tipo de algumas variáveis de "int"(2-bytes) para "int8_t"(1-bytes) tanto globalmente como localmente (devido ao espaço desnecessário de memória reservado).

- Códigos - Comparativo -

- Com a 1ª Implementação -

- Depois da 2ª Implementação -

```
int pairV[MAXN], way[MAXM];

void hungarian() {
    memset(&pairV, 0, sizeof pairV);
    for(int i = 1, j0 = 0; i <= n; i++) {
        pairV[0] = i;
        memset(&minv, INF, sizeof minv);
        memset(&used, false, sizeof used);
        do {
            used[j0] = true;
            int i0 = pairV[j0], j1;
            long delta = INF;
            for(int j = 1; j <= m; j++)
                long cur;
```

```
                for(int j = 1; j <= m; j++) {
                    if (used[j]) {
                        pu[pairV[j]] += delta, pv[j] -= delta;
                    } else {
                        minv[j] -= delta;
                    }
                }
                j0 = j1;
            } while(pairV[j0] != 0);
        } do {
            int j1 = way[j0];
            pairV[j0] = pairV[j1];
            j0 = j1;
        } while(j0);
        for(int i = 1; i < n; i++) {
            int aux = pairV[i + 1];
            pairV[i + 1] = pairV[i];
            pairV[i] = aux;
```

```
void loop() {
    long custo_maximo = 0;
    hungarian();
    for(int j = 1; j <= m; j++) {
        if (pairV[j] == 0) continue;
```

```
int8_t pairV[MAXN], way[MAXM]; // <- Modificado! ("int" -> "int8_t").

void hungarian() {
    memset(&pairV, 0, sizeof pairV);
    for(int8_t i = 1, j0 = 0; i <= n; i++) { // <- Modificado! ("int" -> "int8_t").
        pairV[0] = i;
        memset(&minv, INF, sizeof minv);
        memset(&used, false, sizeof used);
        do {
            used[j0] = true;
            int8_t i0 = pairV[j0], j1;
            long delta = INF;
            for(int8_t j = 1; j <= m; j++)
                long cur;
```

```
                for(int8_t j = 1; j <= m; j++) { // <- Modificado! ("int" -> "int8_t").
                    if (used[j]) {
                        pu[pairV[j]] += delta, pv[j] -= delta;
                    } else {
                        minv[j] -= delta;
                    }
                }
                j0 = j1;
            } while(pairV[j0] != 0);
        } do {
            int8_t j1 = way[j0]; // <- Modificado! ("int" -> "int8_t").
            pairV[j0] = pairV[j1];
            j0 = j1;
        } while(j0);
        for(int8_t i = 1; i < n; i++) { // <- Modificado! ("int" -> "int8_t").
            int8_t aux = pairV[i + 1]; // <- Modificado! ("int" -> "int8_t").
            pairV[i + 1] = pairV[i];
            pairV[i] = aux;
```

```
void loop() {
    long custo_maximo = 0;
    hungarian();
    for(int8_t j = 1; j <= m; j++) { // <- Modificado! ("int" -> "int8_t").
        if (pairV[j] == 0) continue;
```

[Otimização Proposta] - 2ª Implementação - Resultados e Medições -

- Ganhos e Perdas (%) -

- Redução de 0,23% no tempo de computação p/ 1 chamada e Aumento de 8,18% p/1000 chamadas.
- Redução de 8,41% na memória de dados e de 0,59% na memória de código.

Código	Memória de Código < em bytes >	Memória de Dados < em bytes >	Tempo de Computação (p/ 1 chamada do alg) < em microsegundos (μ) >	Tempo de Computação Médio (p/ 1000 chamadas do alg) < em microsegundos (μ) >
< Arduino Uno > Implementação anterior (1ª)	8128	1070	495896 (~0,5 segundos)	491088 (~0,49 segundos)
< Arduino Uno > Após a 2ª Implementação	8080	980	494752 (~0,5 segundos)	531260 (~0,53 segundos)

() [] []
[] []
[] [] []
[] [] @

- “%” de Ganhos e Perdas calculadas pela ferramenta online:
<https://www.calculer.net/financeiro/calculadora-de-porcentagem-calculo-online/>

[Otimização Proposta] - 3ª Implementação - Explicação -

- Explicação -

- Mover as variáveis "globais" que são somente usadas pela função "hungarian()" para dentro dela tornando-as "locais".

- Códigos - Comparativo -

- Com a 2ª Implementação -

- Depois da 3ª Implementação -

```
long pu[MAXN], pv[MAXN];  
long minv[MAXM];  
bool used[MAXM];  
}  
  
int8_t pairV[MAXN], way[MAXM];  
  
void hungarian() {  
  
    memset(&pairV, 0, sizeof pairV);  
    for(int8_t i = 1, j0 = 0; i <= n; i++) {  
        pairV[0] = i;  
        memset(&minv, INF, sizeof minv);  
        memset(&used, false, sizeof used);  
    }
```



```
int8_t pairV[MAXN]; <<< // Não mudamos essa para local pois é utilizada tanto  
                        // pela função "hungarian()" quanto pela "loop()".  
  
void hungarian() {  
  
    long pu[MAXN], pv[MAXN]; // <- Modificado! (Variável "Global" -> "Local").  
    long minv[MAXM]; // <- Modificado! (Variável "Global" -> "Local").  
    bool used[MAXM]; // <- Modificado! (Variável "Global" -> "Local").  
}  
  
    int8_t way[MAXM]; // <- Modificado! (Variável "Global" -> "Local").  
  
    memset(&pairV, 0, sizeof pairV);  
    for(int8_t i = 1, j0 = 0; i <= n; i++) {  
        pairV[0] = i;  
        memset(&minv, INF, sizeof minv);  
        memset(&used, false, sizeof used);  
    }
```


[Otimização Proposta] - 3ª Implementação - Resultados e Medições -

- Ganhos e Perdas (%) -

- **Aumento** de **36,27%** no tempo de computação p/ 1 chamada e de **26,46%** p/1000 chamadas.
- **Redução** de **64,29%** na memória de dados e **Aumento** de **4,28%** na memória de código.

Código	Memória de Código < em bytes >	Memória de Dados < em bytes >	Tempo de Computação (p/ 1 chamada do alg) < em microsegundos (μ) >	Tempo de Computação Médio (p/ 1000 chamadas do alg) < em microsegundos (μ) >
< Arduino Uno > Implementação anterior (2ª)	8080	980	494752 (~0,5 segundos)	531260 (~0,53 segundos)
< Arduino Uno > Após a 3ª Implementação	8426	350	674208 (~0,67 segundos)	671815 (~0,67 segundos)

() [] []
[] []
[] [] []
[] []

- “%” de Ganhos e Perdas calculadas pela ferramenta online:
<https://www.calcul.net/financeiro/calculadora-de-porcentagem-calculo-online/>

[Otimização Proposta] - 4ª e Última Implementação - Explicação -

- Explicação -

- Mover a variável "pairV[MAXN]" de "global" para "local" dentro da função "hungarian()" junto de todas as partes na função "loop()" que usam essa variável para dentro da função "hungarian()" também, deixando-a como um "método só" que já retorna o resultado esperado sem necessitar de outras funções externas para terminar seus cálculos/operações restantes, como método "loop()" fazia anteriormente.

- Códigos - Comparativo -

- Com a 3ª Implementação -

```
int8_t pairV[MAXN];  
  
void hungarian() {  
  
    long pu[MAXN], pv[MAXN];  
    long minv[MAXM];  
    bool used[MAXM];  
  
    void loop() {  
  
        long custo_maximo = 0;  
        hungarian();  
        for(int8_t j = 1; j <= m; j++) {  
            if (pairV[j] == 0) continue;  
            if ((j < m) or (pairV[j] < m)) custo_maximo += (int)pgm_read_word_near(&cost[pairV[j]][j]);  
        }  
        custo_maximo = -custo_maximo;  
  
        mySerial.println(F("Custo maximo da matrix solicitada: "));  
        mySerial.print(custo_maximo);  
        mySerial.println(F("\n"));  
  
        while(true);  
    }  
}
```

- Depois da 4ª e Última Implementação -

```
long hungarian() { // <- Modificado! (Função "hungaria  
///////////////////////////////////////////////////////////////////: <- Modificad  
bool used[MAXM];  
  
int8_t pairV[MAXN], way[MAXM]; // <- Modificado! (Var  
long pv[MAXN], pu[MAXN], minv[MAXM];  
////////////////////////////////////////////////////////////////  
long custo_maximo = 0; // <- Modificado! (Variável  
_  
    pairV[i + 1] = pairV[i];  
    pairV[i] = aux;  
}  
///////////////////////////////////////////////////////////////////: <- Modificado!  
for(int8_t j = 1; j <= m; j++) {  
    if (pairV[j] == 0) continue; //  
    if ((j < m) or (pairV[j] < m)) custo_maximo += (int)pgm_read_word_near(&cost[pairV[j]][j]); //  
}  
//////////////////////////////////////////////////////////////////  
return -custo_maximo; // <- Modificado! (Retorna um long que é o resultado esperado pelo método)  
}  
  
void loop() {  
  
    mySerial.println(F("Custo maximo da matrix solicitada: "));  
    mySerial.println(hungarian()); // <- Modificado!  
    //(Agora, chamar "hungarian()" já retorna em uma linha só o resultado esperado!)  
}
```

[Otimização Proposta] - 4ª e Última Implementação - Resultados e Medições -

- Ganhos e Perdas (%) -

- Aumento de 12,52% no tempo de computação p/ 1 chamada e de 12,55% p/1000 chamadas.
- Redução de 62,86% na memória de dados e de 9,68% na memória de código.

Código	Memória de Código < em bytes >	Memória de Dados < em bytes >	Tempo de Computação (p/ 1 chamada do alg) < em microsegundos (μ) >	Tempo de Computação Médio (p/ 1000 chamadas do alg) < em microsegundos (μ) >
< Arduino Uno > Implementação anterior (3ª)	8426	350	674208 (~0,67 segundos)	671815 (~0,67 segundos)
< Arduino Uno > Após a 4ª (Última) Implementação	7610	130	758616 (~0,76 segundos)	756139 (~0,75 segundos)

() [] []
[] []
[] [] []
[] []

- “%” de Ganhos e Perdas calculadas pela ferramenta online:
<https://www.calcul.net/financeiro/calculadora-de-porcentagem-calculo-online/>

[Quadro Comparativo] - Entre Plataformas - Arduino Uno -

- Ganhos e Perdas (%) - Código Arduino Não-Implementado vs. Código Arduino Implementado -
 - Aumento de 36,28% no tempo de computação p/ 1 chamada e de 33,15% p/1000 chamadas.
 - Redução de 89,60% na memória de dados e de 8,42% na memória de código.

Código	Memória de Código < em bytes >	Memória de Dados < em bytes >	Tempo de Computação (p/ 1 chamada do alg) < em microsegundos (μ) >	Tempo de Computação Médio (p/ 1000 chamadas do alg) < em microsegundos (μ) >
< Arduino Uno > Sem Implementação	8310	1250	556680 (\sim =0,56 segundos)	567906 (\sim =0,57 segundos)
< Arduino Uno > 4ª (Última) Implementação	7610	130	758616 (\sim =0,76 segundos)	756139 (\sim =0,75 segundos)

() [] []
[] []
[] [] []
[] []

- “%” de Ganhos e Perdas calculadas pela ferramenta online:
<https://www.calculer.net/financeiro/calculadora-de-porcentagem-calculo-online/>

[Quadro Geral] - Resultados e Medições -

Código	Memória de Código < em bytes >	Memória de Dados < em bytes >	Tempo de Computação (p/ 1 chamada do alg) < em microsegundos (μ) >	Tempo de Computação Médio (p/ 1000 chamadas do alg) < em microsegundos (μ) >
< Desktop - Linux > Sem Implementação	2630	9784	(\approx 0 Segundos)	287 (\approx 0,000287 segundos)
< Arduino Uno > Sem Implementação	8310	1250	556680 (\approx 0,56 segundos)	567906 (\approx 0,57 segundos)
< Arduino Uno > 1ª Implementação	8128	1070	495896 (\approx 0,5 segundos)	491088 (\approx 0,49 segundos)
< Arduino Uno > 2ª Implementação	8080	980	494752 (\approx 0,5 segundos)	531260 (\approx 0,53 segundos)
< Arduino Uno > 3ª Implementação	8426	350	674208 (\approx 0,67 segundos)	671815 (\approx 0,67 segundos)
< Arduino Uno > 4ª (Última) Implementação	7610	130	758616 (\approx 0,76 segundos)	756139 (\approx 0,75 segundos)

[Aprendizado] - No contexto de SEMBS -

- O que aprendemos -

- Trabalhar a precisão de nossas medições em aplicações embarcadas: Os métodos, técnicas e o estudo por trás de uma boa e precisa medição de um código que utiliza dos recursos limitados de certo microcontrolador(aqui sendo o Atmega328p do Arduino Uno).
- Nem sempre o MAIS COMPLEXO é o MAIS ADEQUADO para se otimizar determinado fator de um código(Memória ou Tempo de Computação), principalmente se for em plataformas microcontroladoras. Seguir as boas práticas de programação que determinada plataforma exige pode ser o suficiente para ganhar um bom tempo de computação ou de espaço de memória.
- Entender a estrutura do código é essencial, dependendo de como ele é organizado grande pode ser o impacto computacional que aplicação gera ao microcontrolador que terá de interpretá-la e executá-la em determinado circuito.

```
( ) [ ] [ ]  
[ ] [ ]  
[ ] [ ] [ ]  
[ ] [ ]
```

...

-



*REFERÊNCIAS - Otimizações - Arduino Uno

- <https://learn.adafruit.com/memories-of-an-arduino?view=all>
- <https://www.circuitbasics.com/how-to-optimize-arduino-code/>
- <https://mcuoneclipse.com/2013/04/14/text-data-and-bss-code-and-data-size-explained/>
- <https://arduino.stackexchange.com/questions/763/im-using-too-much-ram-how-can-this-be-measured>
- <https://stackoverflow.com/questions/30349288/can-long-int-be-returned-by-a-function-in-c>
- <https://arduino.stackexchange.com/questions/533/what-is-the-difference-between-declaring-a-variable-outside-of-loop-and-declaring>

```
( ) [ ] [ ]  
[ ] [ ]  
[ ] [ ] [ ]  
[ ] [ ]
```

*REFERÊNCIAS - Medições - Arduino Uno

-<https://www.calculer.net/financeiro/calculadora-de-porcentagem-calculo-online/>
 -https://www.google.com/search?ei=yxE9YLnIG6qp5OUPrPqaCA&q=microssegundos+para+segundos&oq=microssegundos+para+segundos&gs_lcp=Cqndnd3Mtd2l6EAMyAggAMqYIABAWEB46BwgAEecQsAM6CAgAEBYQChAeULLwWViN_Ilq0IFaaAFwAngAgAGKBYgBhiCSAQoyLTExLjluMC4xmAEAoAEBqgEHZ3dzLXdpesgBCLgBAsABAQ&sclient=gws-wiz&ved=0ahUKEwj52ZrGvY_vAhWqFLkGHSy9BgEQ4dUDCA0&uact=5

-----Referências:

-<https://forum.arduino.cc/index.php?topic=594697.0>
 -<https://www.arduino.cc/reference/en/language/functions/time/micros/>
 -<https://www.arduino.cc/reference/pt/language/functions/time/millis/>
 -<https://oscarliang.com/check-ram-memory-usage-arduino-optimization/>
 -<https://www.codeproject.com/Tips/753716/Howto-find-out-how-much-RAM-your-Arduino-program-u>

() [] []
 [] []
 [] [] []
 [] []

*REFERÊNCIAS - Medições - PC - Windows - Código em C -

- <https://stackoverflow.com/questions/31217181/size-command-in-unix>
- <http://www.cplusplus.com/reference/ctime/clock/>

```
( ) [ ] [ ]  
[ ] [ ]  
[ ] [ ] [ ]  
[ ] [ ]
```

*Duvidas?!



() [] []
[] []
[] [] []
[] []

*Obrigado !!!

;)



() [] []
[] []
[] [] []
[] []

*Sugestão da equipe para música a ser tocada no próximo semestre nas aulas de SEMB:



*Música sugerida: “Chão de Giz”.
(na verdade todas as músicas dele são ótimas kkkk)

*Artista: Zé Ramalho.

**Obs. kkkkkkkk, brincadeira professor nós agradecemos a paciência o tempo e os conselhos dados em nossa apresentação sucesso pra você e se Deus quiser que não nos vejamos mais nessa cadeira de SEMBS (por motivos de passar nela é claro kkkkk).

() [] []
[] []
[] [] []
[] []