

ACH2023 ALGORITMOS E ESTRUTURAS DE DADOS I

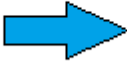
Semestre 2019-2 - Exercício prático 2 – Listas cruzadas – t.02

Estagiário PAE: Samuel Caetano da Silva (samuel.caetano.silva@usp.br)

1. O objetivo do trabalho é implementar de forma correta e completa a função *substituir* em uma matriz de listas cruzadas utilizando apenas listas ligadas de implementação dinâmica como estrutura de armazenamento auxiliar se necessário, conforme modelo fornecido. A assinatura da função é a seguinte:

```
void substituir(LISTASCR* m, ini i, int j)
```

2. A função recebe como entrada uma matriz esparsa quadrada de 10 x 10 dimensões representada em listas cruzadas contendo inteiros maiores do que zero, um número de linha i e um número de coluna j , ambos garantidamente válidos. O objetivo da função é o de trocar todos os elementos (existentes ou não) da linha i pelos elementos da coluna j , e vice-versa, até o limite da intersecção – mas tomando cuidado de não substituir o elemento $[i, j]$ propriamente dito.
3. Exemplo: para o ponto de intersecção d $[4, 3]$ em amarelo, ocorre a troca de todos os elementos azuis (da linha 4) com os elementos verdes (da coluna 3) que antecedem a intersecção d, sem afetá-la.



	1	2	3	4	5
1	a		i		
2		b	j		g
3		c			f
4		k	d	e	
5					h

	1	2	3	4	5
1	a				
2		b	k		g
3		c			f
4		i	j	d	e
5					h

4. Note que o elemento na posição $[i, j]$ em amarelo nunca é alterado, e que neste exemplo específico nem o elemento acima dele sofreu alteração, pois só havia dois elementos possíveis (em azul) na linha 4.
5. Restrições de implementação:
 - Não use nenhum vetor na sua implementação além dos que já fazem parte da matriz. Se necessitar de estruturas auxiliares, use sempre listas ligadas de implementação dinâmica usando o próprio tipo NO já definido para a matriz (lembre-se de que você pode ignorar os campos que não quiser utilizar em uma lista simples).
 - Não defina variáveis globais.
 - Não exiba nenhuma mensagem na tela, nem solicite que o usuário pressione nenhuma tecla etc.
6. A função implementada deve estar inteiramente contida em um arquivo de nome **trabalho.cpp**. Note no entanto que para testar a sua implementação e garantir sua correção você provavelmente terá de criar várias outras funções auxiliares (e.g., entrada de dados, exibição etc.) que não serão avaliadas e que não devem ser entregues.
7. O EP pode ser desenvolvido individualmente ou pelas **mesmas duplas** do EP anterior. Novas duplas não serão aceitas. O link a seguir é fornecido apenas para consulta aos números dos grupos, mas atualizações não serão consideradas. Caso uma dupla decida realizar EPs individuais, basta entregar dois trabalhos separados. O link é acessível usando seu email USP:

<https://docs.google.com/spreadsheets/d/1wZHIZHKaMf1htc4DfbidXG09FkdhfHzn9rmvLZFeI/edit?usp=sharing>

Importante: anote o número do seu grupo para informá-lo no código a ser entregue.

O que/como entregar:

- O programa deve ser compilável no Codeblocks 13.12 sob Windows 7 ou superior. Será aplicado um **desconto de 50%** na nota do EP caso ele não seja **prontamente** compilável nesta configuração.
- Entregue um arquivo trabalho.cpp (exatamente com este nome, sem compactação) contendo a função do EP e todas as rotinas que ela invoca.
- A entrega será via *upload* no sistema Tidia por **apenas um** integrante de cada dupla.
- Preencha no código as declarações *nroUSP1*, *nroUSP2* e *grupo* para que você seja identificado. Se o EP for individual, mantenha o valor do segundo nro. como zeros.

Prazos:

O EP deve ser depositado no prazo definido na atividade cadastrada no sistema Tidia. Não serão aceitos EPs entregues depois do prazo, independentemente do motivo. Entregas no último dia são assim por conta e risco do aluno, e nenhum tipo de imprevisto de última hora (e.g., problemas de saúde, indisponibilidade de rede etc.) pode ser usado como justificativa para o atraso. O EP é uma atividade para ser desenvolvida ao longo de várias semanas, não no último dia da entrega.

O sistema Tidia envia um email de confirmação da submissão. É responsabilidade do aluno que fez o *upload* do arquivo verificar se o mesmo foi corretamente recebido pelo sistema (ou seja, sugere-se fazer *download* novamente para ter certeza). Atrasos/falhas na submissão invalidam o trabalho realizado. Certifique-se também de que a versão entregue é a correta *antes* do prazo final. Não serão aceitas substituições.

Avaliação:

O objetivo do exercício é o de realizar codificações **corretas**, que não necessariamente precisam ser eficientes ou “elegantes”. Ou seja, basta que o programa retorne a matriz correta para cada entrada fornecida. Não existe assim solução “meio” correta: ou a função faz o que foi proposto, ou não faz. Erros de execução e de alocação de memória (muito comuns!) também invalidam o teste, assim como a ausência de eventuais funções auxiliares necessárias para a execução do programa.

O EP será testado com uma série de 10 chamadas fornecendo matrizes com diferentes parâmetros de entrada. Cada teste sem erro de execução que retornar a matriz resultante esperada vale um ponto. Para qualquer resultado diferente do esperado, passa-se ao próximo teste sem pontuar.

Este EP deve ser desenvolvido obrigatoriamente por *todos* os alunos de AED1. Sua nota é parte integrante da média final da disciplina e *não é* passível de substituição. Problemas com EPs – principalmente erros de execução e plágio – são a principal causa de reprovação em ACH2023. Não tente emprestar sua implementação para outros colegas, nem copiar deles, pois isso invalida o trabalho de todos os envolvidos.

Não está funcionando? Use comandos *printf* para saber por onde seu programa está passando e os valores atuais das variáveis, ou os recursos de *debug* do *CodeBlocs*. O propósito do exercício é justamente o de encontrar erros por conta própria – não espere ajuda para isso. Bom trabalho!