Trabalho Prático 3

Descrição do problema

Na cidade de nlogônia as pessoas possuem vários *hobbies* ligados a matemática, um muito comum entre as crianças é aplicar transformações lineares a pontos desenhados em uma folha de papel. A brincadeira funciona da seguinte forma: As transformações serão aplicadas em n instantes de tempo diferentes, e cada instante de tempo está associado a uma transformação linear. Além disso, os pontos "nascem" em um instante de tempo t_0 e "morrem" em um instante de tempo t_d (que pode ser igual a t_0!). A cada instante de tempo em que estão vivos, os pontos são afetados pela transformação linear correspondente àquele instante de tempo. O objetivo final da brincadeira é determinar a posição em que o ponto morre, dado que você sabe o tempo e a posição onde ele nasce, o tempo em que ele morre e as transformações que serão aplicadas a ele. Para deixar tudo mais divertido, as crianças nlogonenses decidiram que as transformações poderão ser alteradas ao longo da brincadeira, desde que seja uma de cada vez, e que inicialmente todos os instantes de tempo estarão associados a transformação identidade. As crianças não gostam de negatividade, então todos os números utilizados serão inteiros positivos! Como a sequência de transformações pode levar os pontos para coordenadas muito grandes, elas estão interessadas somente nos últimos 8 dígitos das coordenadas. Seu trabalho é ajudar as crianças a computar as posições dos pontos enquanto elas brincam.

Formalizando o problema

Inicialmente sabemos que é possível associar transformações lineares a matrizes. Dessa forma dado n instantes de tempo, temos uma sequência $A_1, A_2, \ldots, A_n$ de n matrizes 2×2 (pois os pontos são desenhados em uma folha de papel), em que a matriz A_i será aplicada aos pontos no instante de tempo i. Inicialmente todas essas matrizes são I_2. A partir de agora as crianças irão executar operações. Essas operações podem ser de dois tipos:

1. Atualização: as crianças escolhem um instante de tempo i e uma matriz B válida para as restrições da brincadeira e substituem a matriz A_i pela B.

2. Consulta: Nessa operação as crianças escolhem instantes t_0 e t_d representando respectivamente os instantes de "nascimento" e "morte" de um ponto e as coordenadas x e y que o ponto possuía quando nasceu. O objetivo dessa operação é determinar as coordenadas x e y que o ponto terá quando desaparecer.

Abordagem ineficiente

Uma solução ingenua seria armazenar cada uma das transformações em um arranjo indexado pelos instantes de tempo. Para a operação de atualização bastaria modificar a matriz referente ao instante i, e como o tamanho da matriz não depende da entrada, isso pode ser feito em tempo constante.

Para a segunda operação, supondo que os instantes são respectivamente a e b, bastaria multiplicar todas as matrizes entre a e b e depois aplicar o resultado ao ponto. No pior caso essa abordagem tem complexidade de tempo $\mathcal{O}(n)$. No entanto as crianças nlogonenses amam esta brincadeira, e elas sempre executam **muitas** operações toda vez que brincam. Computar dessa forma iria levar tempo demais e as crianças ficariam impacientes.

Árvore de segmentação

A solução é idealizar uma estrutura de dados capaz de computar essa sequência de operações de forma eficiente. Dessa vez a estrutura escolhida será a árvore de segmentação, ou *segtree*. A ideia é pré-computar a solução para alguns subarranjos, a fim de ser capaz de executar as operações tanto de consulta quanto de atualização rapidamente.

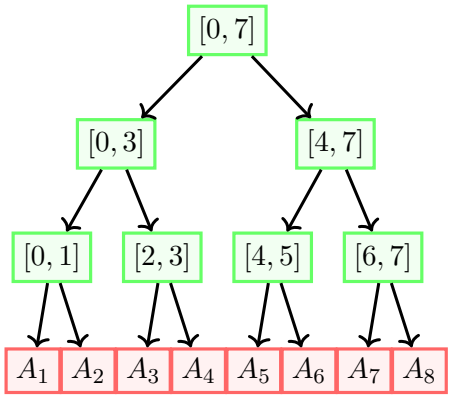

Figura 1: Representação visual da árvore de segmentação. Cada nó será responsável por um segmento do arranjo original e nele computaremos a solução para aquele segmento

Para implementar essa estrutura iremos codificar três funções: Construir uma *segtree*, realizar as consultas e as atualizações. As ideias das implementações se assemelham bastante a outros tipos de árvores binárias que vimos no curso. Os detalhes da implementação de uma *segtree* genérica estão bem descritos no vídeo abaixo.

https://www.youtube.com/watch?v=OW_nQN-UQhA&ab_channel=MaratonaUFMG

O que você deve implementar

Entrada e saída

As instancias serão lidas pela pela entrada padrão. Essa leitura será feita da seguinte forma:

Primeira linha contendo dois inteiros n e q, indicando respectivamente quantos instantes de tempo e quantas operações serão realizadas. As próximas linhas irão descrever as operações que devem ser realizadas na ordem em que foram lidas.

- Atualização: Uma linha contendo o caractere "u" para indicar a operação de atualização seguido de um inteiro a, que indica qual instante de tempo deve ter sua transformação alterada. As próximas duas linhas contém 2 inteiros cada, descrevendo a nova transformação. A operação de atualização não possui nenhuma saída esperada.

- Consulta: Uma linha contendo o caractere "q" para indicar a operação de consulta seguida de quatro inteiros t_0, t_d, x e y, indicando respectivamente os instantes de "nascimento" e "morte" do ponto e suas coordenadas no plano. Essa operação espera como saída dois inteiros separados por um espaço em branco, indicando as coordenadas em que o ponto desapareceu. Apenas os 8 dígitos menos significativos devem ser impressos.

Exemplos

Exemplo 1:

```
3 5
u 0
0 1
0 0
u 1
0 0
0 1
u 2
0 1
0 0
q 0 1 0 1
q 1 2 0 1
```

Saída esperada:

```
1 0
0 0
```

Exemplo 2:

```
3 6
u 0
0 1
1 0
q 0 2 5 7
u 2
0 1
1 0
q 0 2 5 7
u 2
42 397
1048 32
q 0 2 5 7
```

Saída esperada:

```
7 5
5 7
5464 2989
```

Exemplo 3:

```
4 6
u 0
0 1
99999999 0
u 1
0 1
99999999 0
u 2
0 1
99999999 0
u 3
0 1
99999999 0
q 0 3 1 0
q 0 3 0 1
```

Saída esperada:

```
1 0
0 1
```

Exemplo 4:

```
2 4
u 0
17 0
0 17
u 1
5882353 0
0 5882353
q 0 1 1 0
q 0 1 0 1
```

Saída esperada:

```
1 0
0 1
```

Documentação

A documentação do trabalho deve ser entregue em formato **PDF** e também **DEVE** seguir o modelo de relatório que será postado no minha.ufmg. Além disso, a documentação deve conter **TODOS** os itens descritos a seguir **NA ORDEM** em que são apresentados:

1. **Capa:** Título, nome, e matrícula.
2. **Introdução:** Contém a apresentação do contexto, problema, e qual solução será empregada.
3. **Método:** Descrição da implementação, detalhando as estruturas de dados, tipos abstratos de dados (ou classes) e funções (ou métodos) implementados.
4. **Análise de Complexidade:** Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
5. **Estratégias de Robustez:** Contém a descrição, justificativa e implementação dos mecanismos de programação defensiva e tolerância a falhas implementados.
6. **Análise Experimental:** Apresenta os experimentos realizados em termos de desempenho computacional e localidade de referência, assim como as análises dos resultados.
7. **Conclusões:** As Conclusões devem conter uma frase inicial sobre o que foi feito no trabalho, seguido de um sumário das lições aprendidas.
8. **Bibliografia:** Contém fontes utilizadas para realização do trabalho. A citação deve estar em formato científico apropriado que deve ser escolhido por você.
9. Número máximo de páginas incluindo a capa: 10

A documentação deve conter a descrição do seu trabalho em termos funcionais, dando foco nos algoritmos, estruturas de dados e decisões de implementação importantes durante o desenvolvimento.

Evite a descrição literal do código-fonte na documentação do trabalho.

Dica: Sua documentação deve ser clara o suficiente para que uma pessoa (da área de Computação ou não) consiga ler, entender o problema tratado e como foi feita a solução.

Como será feita a entrega

Você deve utilizar a linguagem C ou C++ para o desenvolvimento do seu sistema. O uso de estruturas pré-implementadas pelas bibliotecas-padrão da linguagem ou terceiros é terminantemente vetado. Você DEVE utilizar a estrutura de projeto abaixo junto ao Makefile:

```
– TP
    |– src
    |– bin
    |– obj
    |– include
    Makefile
```

A pasta **TP** é a raiz do projeto; a pasta **bin** deve estar vazia; **src** deve armazenar arquivos de código (*.c, *.cpp, ou *.cc); a pasta **include**, os cabeçalhos (headers) do projeto, com extensão *.h, por fim a pasta **obj** deve estar vazia. O Makefile deve estar na raiz do projeto. A execução do Makefile deve gerar os códigos objeto *.o no diretório **obj** e o executável do TP no diretório **bin**. O arquivo executável **DEVE** se chamar **tp3.out** e deve estar localizado na pasta **bin**. O código será compilado com o comando:

```
make all
```

O seu código será avaliado através de uma **VPL** que será disponibilizada no minha.ufmg. Você também terá à disposição uma VPL de testes para verificar se a formatação da sua saída está de acordo com a requisitada. A VPL de testes não é valorada e não conta como trabalho entregue. Um pdf com instruções de como enviar seu trabalho para que ele seja compilado corretamente estará disponível no minha.ufmg.
A documentação será entregue em uma atividade separada designada para tal no minha.ufmg. A entrega deve ser feita em um único arquivo com extensão .pdf, nomeado no formato nome_sobrenome_matricula.pdf, onde nome, sobrenome e matrícula devem ser substituídos por suas informações pessoais.

Avaliação

O trabalho será avaliado de acordo com os seguintes critérios:

- Definição e implementação das estruturas de dados e funções - (30% da nota total)
- Corretude na execução dos casos de teste - (20% da nota total)
- Apresentação da análise de complexidade das implementações - (15% da nota total)
- Estrutura e conteúdo exigidos para a documentação - (20% da nota total)
- Indentação, comentários do código fonte e uso de boas práticas - (10% da nota total)
- Cumprimento total da especificação - (5% da nota total)

Se o programa submetido não compilar[1], seu trabalho não será avaliado e sua nota será 0. Trabalhos entregues com atrasos sofrerão penalização de 2^{d-1} pontos, com d = dias de atraso.

[1]Entende-se por compilar aquele programa que, independente de erros no Makefile ou relacionados a problemas na configuração do ambiente, funcione e atenda aos requisitos especificados neste documento em um ambiente Linux.

Considerações finais

1. Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.
2. Leia atentamente o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.
3. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.
4. Plágio é crime. Trabalhos onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas (em relação a todos os envolvidos). Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na seção de bibliografia da documentação. **Cópia e compartilhamento de código não são permitidos.**

FAQ (Frequently asked Questions)

1. Posso utilizar alguma estrutura de dados do tipo Queue, Stack, Vector, List, e etc..., do C++? NÃO
2. Posso utilizar smart pointers? NÃO.
3. Posso utilizar o tipo String? SIM.
4. Posso utilizar o tipo String para simular minhas estruturas de dados? NÃO
5. Posso utilizar alguma biblioteca para tratar exceções? SIM.
6. Posso utilizar alguma biblioteca para gerenciar memória? SIM.
7. As análises e apresentação dos resultados são importantes na documentação? SIM.
8. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? NÃO
9. Posso fazer o trabalho em dupla ou em grupo? NÃO
10. Posso trocar informações com os colegas sobre a teoria? SIM.
11. Posso fazer o trabalho no Windows, Linux, ou MacOS? SIM.
12. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? SIM.