MC202 — ESTRUTURAS DE DADOS

Laboratório 08 — Árvore binária de busca

Tarefa

A aliança rebelde está com uma tática nova de combate onde são organizadas naves identificadas em uma fileira ordenadas do menor identificador para o maior identificador. O número de naves é sempre da forma $2^k - 1$, onde k é um inteiro tal que $k \geq 1$. As naves são identificadas unicamente através de um inteiro. Veja Figura 1.

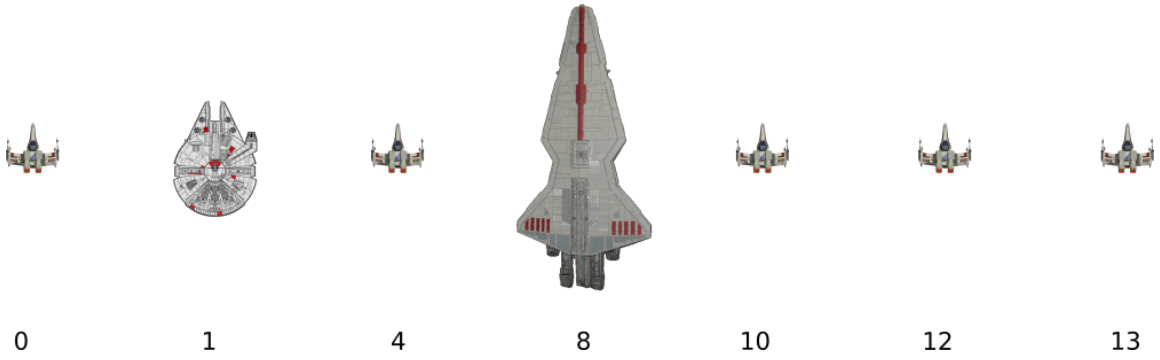

Figura 1: Naves organizadas em fileira identificadas unicamente por um inteiro.

Para entrar em modo de combate, as naves decolam e ficam em formato de triângulo de ponta-cabeça, como é visto na Figura 2.

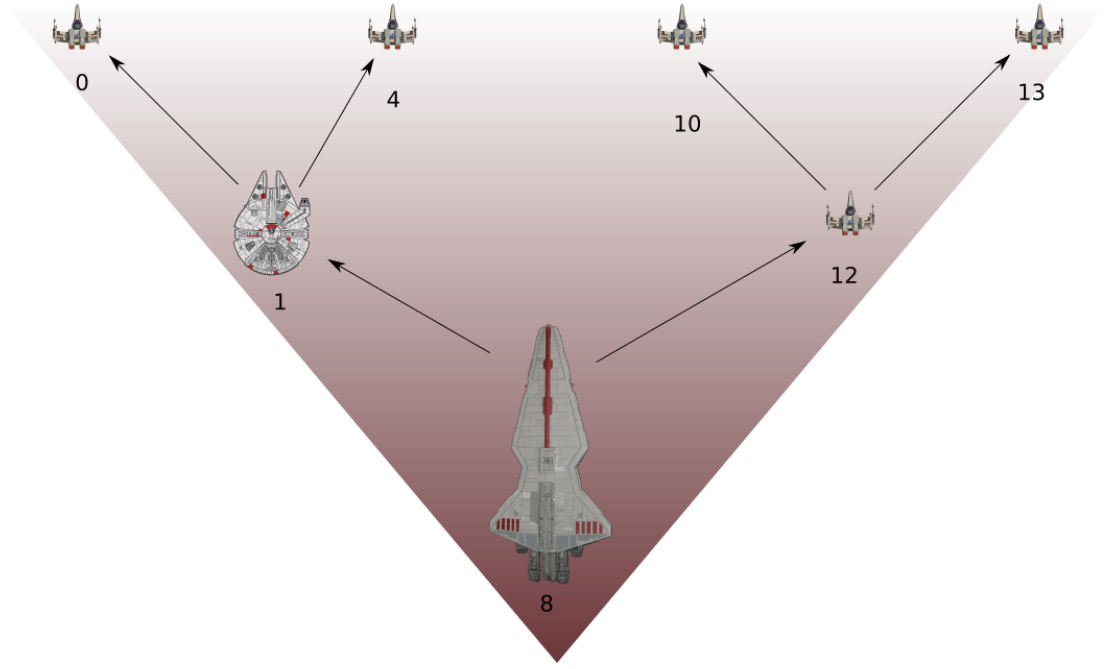

Figura 2: Posição de combate das naves para $k = 3$.

Essa formação tem as seguintes propriedades:

1. as posições iniciais formam uma árvore binária de busca completa
2. toda nave comanda dois esquadrões de naves, um à esquerda e outro à direita
3. toda nave em esquadrão esquerdo tem identificador menor que de seu comandante
4. toda nave em esquadrão direito tem identificador maior que de seu comandante

Uma ilustração dessa formação das pode ser visto na Figura 2, com $k = 3$.

Para avaliar a habilidade dos pilotos durante o combate, uma das informações utilizadas é quantas vezes cada nave foi efetivamente atacada. Para isso, cada nave reporta toda vez que foi atingida.

Como um programador da aliança rebelde, sua tarefa será construir um programa que irá ajudar na formação e na avaliação dos pilotos. O seu programa receberá os identificadores ordenados e o relatório de ataques, então seu programa deve imprimir o identificador e o número de ataques sofridos pela nave de acordo com um percurso pré-ordem na formação.

Entrada

A entrada será composta por um inteiro $N\ (1 \leq N \leq 2^{16})$ que indicará o número de naves, seguido por N inteiros que serão os identificadores das N naves, ordenados do menor identificador para o maior.

Depois, será dado um inteiro $M\ (0 \leq M \leq 10^5)$ que será o número ataques ocorridos durante a batalha, seguido por M inteiros que são os identificadores das naves que foram atingidas por cada ataque.

Saída

A saída será composta por N linhas compostas por dois inteiros e um caracter de final de linha, sendo o primeiro inteiro o identificador da nave e o segundo o número de vezes na qual ela foi atacada. As naves estarão ordenadas de acordo com um percurso pré-ordem na formação remanescente.

Entrada

```
7
0 1 4 8 10 12 13
7
12 8 1 10 1 10 1
```

Saída

```
8 1
1 3
0 0
4 0
12 1
10 2
13 0
```

Entrada

```
15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
3
12 8 11
```

Saída

```
8 1
4 0
2 0
1 0
3 0
6 0
5 0
7 0
12 1
10 0
9 0
11 1
14 0
13 0
15 0
```

Critérios específicos

- Você deverá implementar e utilizar uma estrutura de dados do tipo árvore binária de busca para armazenar os identificadores das naves
- Deverão ser submetidos os seguintes arquivos:
    - arvore.h (sua interface da estrutura árvore)
    - arvore.c (implementação da estrutura árvore)

    - lab08.c (solução do problema do laboratório)

- Tempo máximo de execução: 1 segundo.

Testando

Para compilar usando o Makefile fornecido e verificar se a solução está correta basta seguir o exemplo abaixo.

```
make
./lab03 < arq01.in > arq01.out
diff arq01.out arq01.res
```

onde `arq01.in` é a entrada (casos de testes disponíveis no SuSy) e `arq01.out` é a saída do seu programa. O Makefile também contém uma regra para testar todos os testes de uma vez; nesse caso, basta digitar:

```
make testar_tudo
```

Isso testará o seu programa com os casos abertos. Após o prazo, os casos de teste fechados serão liberados e podem ser baixados digitando-se:

```
make baixar_fechados
```

Para ver quanto tempo seu programa demora em cada teste, digite:

```
make tempo
```

Observações gerais

No SuSy, haverá 3 tipos de tarefas com siglas diferentes para cada laboratório de programação. Todas possuirão os mesmos casos de teste. As siglas são:

1. **DRAFT**: Esta tarefa serve para testar o programa no SuSy antes de submeter a versão final. Nessa tarefa, tanto o prazo quanto o número de submissões são ilimitados, porém arquivos submetidos aqui não serão corrigidos.

2. **ENTREGA**: Esta tarefa tem limite de *uma única submissão* e serve para entregar a versão final dentro do prazo estabelecido para o laboratório. Não use essa tarefa para testar o seu programa: submeta aqui quando não for mais fazer alterações no seu programa.

3. **FORAPRAZO**: Esta tarefa tem limite de *uma única submissão* e serve para entregar a versão final após o prazo estabelecido para o laboratório, mas com nota reduzida (conforme a ementa). O envio nesta tarefa irá substituir a nota obtida na tarefa ENTREGA apenas se o aluno tiver realizado as correções sugeridas no feedback ou caso não tenha enviado anteriormente em ENTREGA.

Observações sobre SuSy:

- Versão do GCC: gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-36)
- Flags de compilação:
  `-std=c99 -Wall -Werror -Werror=vla -pedantic-errors -g -lm`

Além das observações acima, esse laboratório será avaliado pelos critérios gerais:

- Indentação de código e outras boas práticas, tais como:
  - uso de comentários (úteis e apenas quando forem relevantes);
  - código simples e fácil de entender;
  - sem duplicidade (partes que fazem a mesma coisa).
- Organização do código:
  - tipos de dados criados pelo usuário e funções bem definidas e tão independentes quanto possível.
- Corretude do programa:
  - programa correto e implementado conforme solicitado no enunciado;
  - inicialização de variáveis sempre que for necessário;
  - dentre outros critérios.