MC202 — ESTRUTURAS DE DADOS

Laboratório 09 — Escalonamento

Tarefa

Em um computador, os processos, que correspondem aos programas sendo executados, aparentam ser executados simultaneamente, porém um processador só consegue executar um processo por vez. O que acontece é que o processador troca os processos sendo executados tão rapidamente que cria a ilusão de serem simultâneos.

Para decidir qual processo deve ser o próximo a ser executado, existe o *escalonador de processos*, que, de maneira simplificada, considera por quanto tempo um processo está esperando para ser executado. O próximo processo é aquele que estiver esperando há mais tempo.

Uma maneira de manter organizados esses tempos de espera, para que seja fácil a decisão de qual deve ser o próximo processo a ser executado, é usando uma árvore de busca, especificamente uma árvore rubro-negra. Sua tarefa é implementar uma árvore rubro-negra que representa os tempos de espera dos processos, ordenada de maneira crescente. A árvore começa vazia e alguns novos processos podem ser inseridos na árvore durante uma sessão. O objetivo é devolver o processo com maior tempo de espera. Também é possível que o computador tenha mais de um processador, então deve ser devolvido um processo para cada processador disponível.

Entrada

Na primeira linha há um número M ($0 \leq M \leq 1000$) que representa o número de processos a serem inseridos na árvore. Em seguida, há uma linha com M números, cada um, um único número inteiro que representa o tempo de espera de um processo.

Por fim, há uma linha com um número P ($0 \leq P \leq 512$) que representa o número de processadores disponíveis.

Saída

Devem ser impressas P linhas, cada uma com o tempo de espera do processo que será executado no processador P_i. Os processos são distribuídos a partir do primeiro processador P_0, que recebe o processo com maior tempo de espera, P_1, o processo com o segundo maior tempo de espera e assim por diante.

Exemplo

Entrada

```
3
10 2 4 18
1
```

Saída

```
18
```

Entrada

```
10
13 8 17 1 11 15 25 2 6 22
1
```

Saída

```
25
```

Critérios específicos

Descreva os critérios específicos obrigatórios quando houver (qual estrutura usar, nomes dos arquivos, tempo, etc.…)

- Deverão ser submetidos os seguintes arquivos:
    - arvore.c (implementação da árvore rubro negra)
    - arvore.h (interface da árvore rubro negra)
    - lab09.c (programa principal cuja solução utiliza a árvore)
- É obrigatório resolver usando árvore rubro-negra
- Tempo máximo de execução: 5 segundos.

Testando

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

```
make
./lab09 < 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.