INE5452 - Tópicos Especiais em Algoritmos II

Segundo simulado - Questões extra-URI

Entrega: até 30 de setembro de 2020 (até 23:55h via Moodle)

Este Exercício-Programa (EP) é individual. Todos devem entregar as seguintes tarefas (além daquelas disponíveis via sistema externo URI):

- Implementação da *União de Conjuntos* utilizando a estrutura de dados: *Union-Find* com união por *rank*;
  - É desejável que a heurística de *compressão de caminhos* esteja também implementada.
- Implementação de um *Escalonador de Processos* utilizando um *max-heap*.

1 O que deve ser entregue?

O EP pode ser entregue até dia 30 de setembro de 2020 (até 23:55h via Moodle). Cada aluno (aluna) deverá entregar um conjunto de arquivos com:

1. um conjunto de arquivos que implementam a *União de Conjuntos*;
2. um *Makefile* para compilação/interpretação/execução da *União de Conjuntos*;
3. um conjunto de arquivos que implementam um *Escalonador de Processos*;
4. um *Makefile* para compilação/interpretação/execução do *Escalonador de Processos*;
5. cinco casos de teste (independentes entre si) para cada tarefa;
6. uma simulação de cada caso de teste, destacando os diferentes estados da estrutura de dados, a cada passo.

Orientações para construção de um *Makefile*: `https://www.gnu.org/software/make/manual/make.html`

2 Sobre as compilações/interpretações/execuções das tarefas

No momento da execução dos programas desenvolvidos, a presença (virtual) do aluno que o desenvolveu poderá ser necessária para a efetiva avaliação.

3 O que será avaliado na execução/uso do analisador léxico

Abaixo listamos itens importantes com relação a essa avaliação.

- A existência de *Makefiles* (se não existir algum, então a nota das tarefas é 0);
- A execução correta dos *Makefiles* (se algum não executar corretamente, então a nota das tarefas é 0);

- A compilação/interpretação dos programas desenvolvidos (se houver erros de compilação/interpretação, então haverá descontos na nota);
- A existência de cinco casos de testes para cada tarefa (se não houver cinco casos de testes para cada tarefa, então a nota das tarefas é 0).
- A existência das simulações (se não houver alguma, então a nota das tarefas é 0).

4 Sobre a entrada e a saída dos dados

4.1 *União de Conjuntos*

A entrada para a *União de Conjuntos* será um número inteiro n, seguido por n conjuntos de números inteiros, seguido por um m e seguido por m operações de união da forma i j onde i e j são elementos dos conjuntos dados.

Exemplo de entrada:

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

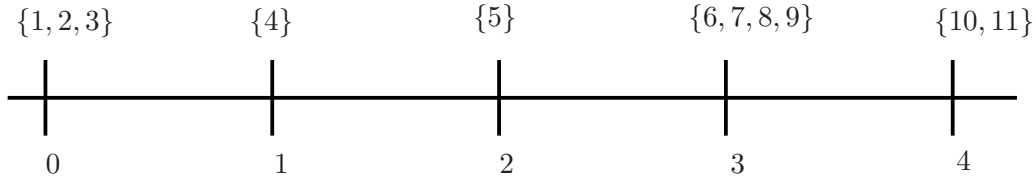
A saída correspondente terá uma linha para cada conjunto resultante (cada linha em ordem lexicográfica assim como as linhas entre si em ordem lexicográfica. Por exemplo, a linha 1 está antes da linha 2 pois a linha 1 começa com `1` e a linha 2 começa com `3`).

Exemplo de saída:

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

4.2 *Escalonador de Processos*

Consideramos que um processo i possui um tempo para processamento $t(i)$, uma prioridade $p(i)$ e um identificador $v(i)$. Um conjunto de processos chega na fila de processos a cada instante de tempo. No exemplo abaixo vemos quatro conjuntos chegando nos tempos 0, 1, 2, 3, e 4.



Na próxima tabela você encontrará o tempo, a prioridade e o identificador de cada processo da figura.

Processo i	Tempo de proc. $t(i)$	Prioridade $p(i)$	Identificador $v(i)$
1	3	5	1
2	3	3	2
3	3	5	3
4	2	1	4
5	2	1	5
6	3	5	6
7	1	5	7
8	2	1	8
9	1	3	9
10	2	2	10
11	4	5	11

Um *max-heap* deverá ser mantido a cada instante de tempo. O sistema mantém em execução o processo na raiz do *heap*. Um processo i entra no *heap* no tempo $t(i)$. Um processo i sai do *heap* somente depois de estar na raiz e depois de terminar o seu tempo de processamento. Dados dois processos i e j, a *prioridade* de i sobre j ($i < j$) valerá quando:

- $p(i) < p(j)$;
- se $p(i) = p(j)$ então $t(i) < t(j)$;
- se $p(i) = p(j)$ e $t(i) = t(j)$ então $v(i) < v(j)$.

Não haverá dois processos com um mesmo identificador. Defina o *tempo de heap* de um processo i como sendo o tempo em que i saiu do *heap* menos o tempo em que i entrou no *heap*.

A entrada dos dados conterá os processos com seus atributos. Primeiro, será dado um valor t que representa o tempo a ser considerado. Em seguida, serão dados t conjuntos de processos (um por linha). Depois, será dado um n que representa o número de processos dados. Finalmente, $4n$ atributos (cada 4 por linha sendo eles i, $t(i)$, $p(i)$, e $v(i)$) A saída deverá conter a ordem de execução dos processos considerando o critério de prioridade descrito acima; o processo que possui o menor tempo de heap (se houver empate, escolha o de menor identificador); o valor do menor tempo de *heap*; o processo que possui *o maior tempo de heap* (se houver empate, escolha o de menor identificador); e o valor do maior tempo de *heap*.

Exemplo de entrada:

```
4
1 2 3
4
5
6 7 8 9
10 11
11
1 3 5 1
2 3 3 2
3 3 5 3
4 2 1 4
```

```
5 2 1 5
6 3 5 6
7 1 5 7
8 2 1 8
9 1 3 9
10 2 2 10
11 4 5 11
```

Saídas correspondentes:

```
Ordem dos processos:  1, 7, 3, 6, 11, 9, 2, 10, 4, 5, 8
Identificador do processo com menor tempo de heap:  7
Valor do menor tempo de heap:  1
Identificador do processo com maior tempo de heap:  8
Valor do maior tempo de heap:  23
```

Observações importantes:

1. Os programas podem ser escritos em C (compatível com compilador gcc versão 5.4.0), C++ (compatível com compilador g++ versão 5.4.0), Java (compatível com compilador javac versão 1.8.0_31) ou Python (compatível com versão 3.7.3), e deve ser compatível com Linux/Unix.

2. Se for desenvolver em Python, então especifique (nos *Makefiles* principalmente) qual é a versão que está usando. Prepare seus *Makefiles* considerando a versão usada.

3. Exercícios-Programas atrasados **não** serão aceitos.

4. Programas com *warning* na compilação terão diminuição da nota.

5. Escreva o seu programa de maneira a destacar a sua formatação.

6. O programa devem começar com um cabeçalho contendo o seu nome e quais pontos são resolvidos.

7. Coloque comentários em pontos convenientes do programa, e faça uma saída clara.

8. A entrega do Exercício-Programa deverá ser feita no Moodle.

9. O Exercício-Programa é individual. Não copie o programa de outro aluno (aluna), não copie os casos de teste de outro aluno (aluna), não empreste o seu programa para outro aluno (aluna), e tome cuidado para que não copiem seu programa sem a sua permissão. Todos os programas envolvidos em cópias terão nota igual a ZERO.

Bom trabalho!