ACH2002 - Introdução à Análise de Algoritmos (2º. Semestre/2022)

Prof.Dra. Ariane Machado Lima

Exercício de Programação 2: **Ordenação, Heap, Algoritmo Guloso (versão: 2)**

Data de entrega: **14/12/2022 - quarta-feira (até 10h da manhã)**

---

1. Introdução

A empresa AG Sistemas quer aproveitar o próximo período de atualizações no sistema principal para colocar em produção a **máxima quantidade** de atualizações possíveis. O período de atualizações ocorrerá durante um final de semana, começando às 22 horas de sábado e terminando às 12 horas de domingo, podendo ser estendido em até 6 horas, um período de **14 a 20 horas** (número inteiro).

O gerente de projetos da AG Sistemas deve informar **quantas** e **quais** atualizações serão incluídas durante o período de atualizações. A lista de atualizações pendentes ("backlog") foi organizada de forma que as atualizações são independentes e possuem o tempo para colocação em produção expresso em horas inteiras. A tabela abaixo é um exemplo.

Tabela 1

Número da Atualização	Duração
1	4
2	3
3	7
4	3
5	6
6	5
7	12
8	4
9	5
10	5

Serão apresentados **dois cenários** distintos. Um primeiro **cenário estático**, no qual não são previstas alterações no planejamento das atualizações. Um segundo **cenário dinâmico**, no qual podem ocorrer inclusões de novas atualizações não previstas inicialmente e também diminuição da duração de alguma atualização previamente planejada.

2. Objetivo

Desenvolver um programa em linguagem C ou java que resolva este problema de otimização enfrentado pelo gerente de projetos da AG Sistemas. O programa deve **obrigatoriamente** utilizar algum algoritmo de **ordenação** para o cenário 1, a estrutura

de dados **Heap** para o cenário 2 (dinâmico) e a técnica de **Algoritmos Gulosos** para ambos os cenários**.**

3. Definições de funcionamento

- O programa **deve obrigatoriamente** aceitar três parâmetros de entrada:
  - **(1)** a duração do período de atualizações que pode ser 14, 15, 16, 17, 18, 19 ou 20 horas;
  - **(2)** o número do cenário executado: 1 ou 2;
  - **(3)** o nome do arquivo de entrada que podem ser **do tipo** entrada1.txt ou entrada2.txt, conforme o cenário;
- Exemplos para o cenário 1 e uma janela de 14 horas, onde **99999999** é o número USP do estudante:
  - `$./ep2_99999999 14 1 entrada1.txt (em C)`
  - `$java ep2_99999999 14 1 entrada1.txt (em java)`.

- **Não** deve ser utilizado arquivo *makefile*;
- O programa fonte em linguagem C ou java **deverá** consistir de um único arquivo chamado **ep2_99999999.c ou ep2_99999999.java**;
- O programa em C ou java deve ser compilável utilizando linha de comando no *prompt* do sistema operacional (cmd.exe no Windows ou terminal no Linux), qualquer opção de compilação deve ser explicitamente informada na entrega;
- O programa deve gerar um arquivo de saída chamado **saida1.txt ou saida2.txt**, conforme o cenário, no mesmo diretório onde o programa é executado;

4. Definição do arquivo de entrada do cenário 1

Primeira e única linha:

- ➔ **campo1**: quantidade de atualizações (**Qa**) inicialmente disponíveis para entrar durante o período de atualizações. Tamanho máximo de atualizações é 100;
- ➔ **<espaço>**
- ➔ **campo2 … campoQa+1**: lista das durações (inteiros) das atualizações 1, 2, 3, ..., Qa respectivamente, separados por **<espaço>;**
- ➔ **<\n>** para finalizar a linha.

Como exemplo, um arquivo de nome entrada1.txt conforme a **tabela 1**.

```
10 4 3 7 3 6 5 12 4 5 5
```

5. Definição do arquivo de entrada do cenário 2

Primeira linha:

- ➔ **campo1**: quantidade de atualizações (**Qa**) inicialmente disponíveis para entrar durante o período de atualizações. Tamanho máximo de atualizações é 100;
- ➔ **<espaço>**
- ➔ **campo2 … campoQa+1**: lista das durações (inteiros) das atualizações 1, 2, 3, ..., Qa respectivamente, separados por **<espaço>;**
- ➔ **<\n>** para finalizar a linha;

Segunda linha em diante (número máximo de operações "i" +"c" é 100):

- ➔ **campo1**: operação de inclusão “i” (caractere i) ou operação de alteração de duração “c” (caractere c);
- ➔ **<espaço>**
- ➔ **campo2**: instante de tempo em que ocorrerá a operação (inclusão ou alteração de duração). Para simplificação será adotado este instante como hora inteira, assim, este instante de tempo varia de 1 até o final do período de atualizações (14 a 20 horas). Ao longo das linhas, esse instante de tempo é sempre apresentado de forma crescente (isto é se t_i e t_j são, respectivamente, os instantes de tempo definidos nas linhas i e j, se i < j então $t_i < t_j$). Uma atualização não pode ser fracionada, portanto, a operação será realizada após a finalização da atualização em execução. *Atenção: está operação pode alterar a escolha das próximas atualizações*;
- ➔ **campo3:** o número da atualização. A **nova atualização** incluída (quando o campo1 = "i") tem número **Qa+1**. Uma operação de alteração de duração mantém a numeração inicialmente estabelecida;
- ➔ **<espaço>**
- ➔ **campo4:** a duração da atualização, incluída ou alterada.
- ➔ **<\n>** para finalizar a linha.

Abaixo um exemplo de arquivo de entrada para este segundo cenário, de tipo entrada2.txt. A primeira linha, conforme a **tabela 1**. Na segunda linha, uma operação de alteração de duração da atualização número 8 (observe na **linha 1** que a atualização **número 8** tem **duração de 4h**), que acontecerá no **instante de tempo 2**, com **nova duração de 3h**. Na terceira linha, uma operação de inclusão de uma nova atualização de número 11 (observe na **linha 1** que a última atualização é número **10**), no **instante de tempo 3**, com **duração de 2h**.

```
10 4 3 7 3 6 5 12 4 5 5
c 2 8 3
i 3 11 2
```

6. **O arquivo saída dos cenários 1 ou 2** tem uma linha com o seguinte formato:
   - ➔ **campo1**: quantidade de atualizações previstas **(Qp)** para entrar no período de atualizações;
   - ➔ **<espaço>**
   - ➔ **campo2 … campoQp+1**: lista de números de atualizações separados por **<espaço>**
   - ➔ **<\n>** para finalizar a linha.

Como exemplo, um arquivo saida1.txt do cenário 1, para o arquivo de entrada entrada1.txt exemplificado anteriormente e um período de atualização definido em 14 horas. **Quatro** atualizações seriam possíveis: número **2** (3h), número **4** (3h), número **1** (4h) e número **8** (4h), as quais ocupam 14 horas do período de atualizações definido.

```
4 2 4 1 8
```

Como exemplo, um arquivo saida2.txt do cenário 2, para o arquivo de entrada entrada2.txt exemplificado anteriormente e um período de atualização estendido para 20

horas. **Seis** atualizações seriam possíveis: número **2** (3h), número **11** (2h), número **4** (3h), número **8** (3h), número **1** (4h) e número **6** (5h), as quais ocupam 20 horas do período de atualizações estendido. Observem que nos instantes de tempo 2 e 3, a atualização número 2 já estava em curso.

```
6 2 11 4 8 1 6
```

Observações - notem que:
1) não necessariamente a soma das durações das atualizações previstas é exatamente igual à duração total do período de atualizações;
2) se duas atualizações possuem a mesma duração elas possuem a mesma prioridade para serem selecionadas.

7. **Orientações**
   - O trabalho é INDIVIDUAL, com o nome do aluno e número USP devidamente identificado nas primeiras linhas do código-fonte.
   - A entrega será realizada em atividade específica para o EP1 no ambiente e-disciplinas, até **14/12/2022 - quarta-feira (até 10h manhã)**.
   - Os cenários 1 e 2 serão corrigidos separadamente, como EP's separados, valendo 10 cada um deles;
   - Deverá ser entregue:
     - O arquivo com o programa-fonte em C ou java.
     - Um arquivo **readme.txt** contendo a linha de comando para execução do programa e a linha de comando de compilação do programa. Além de uma linha com seu Nome completo e número USP.
     - O programa será compilado como abaixo, caso não informado no **readme.txt**:
       - `(C) $gcc -o ep2 ep2_99999999.c`
       - `(java) $javac ep2_99999999.java`
   - Trabalhos com evidências de plágio serão desconsiderados e os autores receberão nota **ZERO**.