Junção de dois arquivos grandes

Uma tarefa comum em análise de dados é realizar a junção de dois, ou mais, arquivos com base em algum de seus campos. Por exemplo, considere os arquivos a seguir:

Arquivo1.txt

```
João      20
Maria     25
José      20
Pedro     20
```

Aquivo2.txt

```
José      1.87
Maria     1.65
```

O arquivo *Arquivo1.txt* contém nome e idade de algumas pessoas. O arquivo *Arquivo2.txt* contém nome e altura de algumas pessoas. Suponha que estejamos interessados em um arquivo que tenha nome, idade e altura de todas as pessoas, ordenadas por nome, que estão tanto em *Arquivo1.txt* quanto em *Arquivo2.txt*, como exemplificado abaixo:

Resultado:

```
José      20   1.87
Maria     25   1.65
```

Como podemos proceder? A essa tarefa dá-se o nome de Junção, a qual será o objeto de implementação desse trabalho.

Problema

O problema introduzido acima fica mais interessante (e complicado) quando os arquivos têm muitos campos, são muito grandes e temos interesse em realizar a junção com base em vários campos. O problema a ser resolvido é formalmente descrito a seguir.

Um arquivo é uma coleção de *N* linhas (sendo *N* possivelmente muito grande). Cada linha do arquivo possui *K strings* separadas pelo caractere '\t' (Tab), as quais são indexadas a partir do 0. A memória principal do computador só é capaz de armazenar *M* linhas do arquivo.

A junção de dois arquivos pode ser feita por um ou mais campos, os quais devem ser especificados por uma lista de números inteiros, *L1* (para o primeiro arquivo) e *L2* (para o segundo arquivo), onde o número de elementos em *L1* deve ser o mesmo de *L2*.

A saída deve ser um novo arquivo. As linhas do novo arquivo são formadas pelas linhas dos dois arquivos de entrada, mas somente aquelas que tenham os campos dados por *L1* do primeiro arquivo iguais as que tenham os campos dados por *L2* do segundo arquivo.

Se o primeiro arquivo possuir *K1* campos, o segundo arquivo *K2* campos e a junção for feita com base em *l* campos, então o novo arquivo terá *K1 + K2 – l* campos, dados por:
Os *l* campos usados na junção em ordem; Os *K1 – l* campos do primeiro arquivo que não estiverem na lista de junção, na mesma ordem que estavam originalmente no arquivo; os *K2 - l* campos do segundo arquivo que não estiverem na lista de junção, na mesma ordem que estavam originalmente no arquivo.

A ordenação das linhas da saída deve respeitar a ordem dos atributos especificados na lista de campos de junção. Em outras palavras, as linhas da saída estarão primeiramente ordenadas pelo primeiro atributo da lista, depois pelo segundo e assim sucessivamente.

Observação: todos os campos serão *strings*, de forma que os algoritmos de ordenação devem trabalhar com *strings* (nesse caso, 10 < 2).

Abaixo, um exemplo:

Arquivo 1 (*N = 10*, *K1 = 4*)

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

Arquivo 2 (*N = 5*, *K2 = 3*)

```
1     a     10
1     b     1
4     b     3
4     b     5
5     5     2
```

Suponha agora que *L1 = 1,0* e *L2 = 0,2*. Então queremos juntar esses dois arquivos comparando a coluna 1 do Arquivo 1 com a coluna 0 do Arquivo 2 e a coluna 0 do arquivo 1 com a coluna 2 do arquivo 2. O arquivo de saída será:

```
1     10    1     1     a
4     3     4     3     b
5     2     4     3     5
```

Repare que as duas primeiras colunas são os campos de junção, as duas próximas colunas vêm do Arquivo 1, sendo a última coluna proveniente do Arquivo 2.

Algoritmo

O problema de junção de dois arquivos pode ser resolvido de uma forma muito simples. **Primeiro**, ordene o primeiro arquivo de acordo com os campos em *L1* (veja que ordenação externa pode ser necessária). **Segundo**, ordene o segundo arquivo de acordo com os campos em *L2* (novamente, você pode precisar de ordenação externa). **Por fim**, faça a intercalação do primeiro arquivo com o segundo arquivo, mantendo apenas as linhas que tenham chaves em *L1* e *L2*, respectivamente, iguais.

Observação: a não ser que *M* seja maior ou igual a *N*, obrigatoriamente você deve usar ordenação direta e externa das linhas de cada arquivo. A violação desse quesito implicará em nota zero.

Entrada

A entrada será feita via linha de comando, onde o nome do executável a ser gerado deve ser *join*. O formato da entrada deve respeitar o seguinte padrão:

```
./join M i1,i2,...,ik j1,j2,...,jk <arquivo1> <arquivo2> <arquivo3>
```

Onde:

- *M* é o número máximo de registros (linhas) que cabem em memória principal
- *i1,i2,...,ik* é a lista *L1*, ou seja, os campos de junção do primeiro arquivo, separados por vírgula
- *j1,j2,...,jk* é a lista *L2*, ou seja, os campos de junção do segundo arquivo, separados por vírgula
- *<arquivo1>* é o nome (ou caminho) do primeiro arquivo
- *<arquivo2>* é o nome (ou caminho) do segundo arquivo
- *<arquivo3>* é o nome (ou caminho) do arquivo de saída, com o resultado da junção.

Por exemplo,

```
./join 1000000 10,0 7,11 fileOne fileTwo fileOut
```

significa que queremos fazer a junção dos arquivos *fileOne* e *fileTwo*. Durante a ordenação externa, é permitido manter até um milhão de registros em memória principal. O resultado da junção deverá estar no arquivo *fieOut*. A junção será feita de forma que o campo 10 de *fileOne* seja igual ao campo 7 de *fileTwo* e o campo 0 de *fileOne* seja igual ao campo 11 de *fileTwo*. Por fim, as linhas do arquivo *fileOut* estarão ordenadas primeiramente pelo campo 10 e depois pelo campo 0 de *fileOne* (equivalentemente, pelos campos 7 e 11 de *fileTwo*).

Observação 1: assim como nos arquivos de entrada, os campos dos registros (linhas) do arquivo de saída devem estar separados por um `'\t'` (Tab).

Observação 2: o caractere `'\t'` será usado unicamente para separação de campos.

Observação 3: você pode assumir que a linha de comando estará sempre formatada de forma correta e seguindo o padrão acima; que os dois arquivos de entrada existem; e que você tem permissão para criar o arquivo de saída.

Observação 4: você pode assumir que cada combinação possível de valores dos campos em *L1* está associada a **no máximo** uma linha do primeiro arquivo e que cada combinação possível de valores dos campos em *L2* está associada a **no máximo** uma linha do segundo arquivo.

Regras:

- O trabalho deve ser implementado em C++ (compilador g++)
- O trabalho pode ser feito grupo de até 4 pessoas (pessoas do mesmo grupo do trabalho 1 não podem estar no mesmo grupo no trabalho 2).
- **Plágio não será tolerado. O regimento acadêmico será seguido à risca em caso de suspeita**

Critérios de correção

Os seguintes critérios serão considerados:

- Se usar variáveis globais, nota zero;
- Se usar goto, nota zero;
- Se seu trabalho não compilar, nota zero. Se por algum motivo seu código compilar (ou funcionar) no seu computador, mas não no meu, o critério de "desempate" será se o seu trabalho compila (funciona) nos computadores do laboratório CCE 416, considerando o Sistema Operacional Ubuntu. Muita atenção aos usuários de **Windows**!
- Os demais critérios de correção são os seguintes:
    - Fração de respostas corretas;
    - Organização e modularização do código;
    - Legibilidade, i.e., código comentado e nomes intuitivos para as variáveis;
    - Presença de vazamento de memória e acesso a posições inválidas de memória. Use **Valgrind** desde os primeiros testes! Em casos extremos, a não observância desse quesito implicará em nota zero.

Entrega

Cada grupo deve enviar apenas um trabalho para o e-mail `gcom.tp.sub@gmail.com`, até às 23:59 do dia **21 de outubro de 2018**. O trabalho deve ser enviado por um e-mail de domínio ufv.br (e-mails de qualquer outro domínio não serão considerados).

O título (*subject*) do e-mail deve conter apenas o número de matrícula dos integrantes do grupo (sem o ES), separados por uma vírgula. Se você optar for fazer o trabalho sozinho, esse campo terá apenas seu número de matrícula.

Se algum grupo enviar mais de um trabalho, apenas o e-mail mais recente será considerado.
O e-mail deve ter em anexo apenas um arquivo comprimido, no formato `.tar.gz`, com nome `trab2.tar.gz`, contendo seu código fonte e um `Makefile`.

Após baixar o arquivo, o professor digitará os comandos:

```
tar -xzvf trab2.tar.gz
make
```

Após esses dois comandos, deve ser gerado um executável de nome `join`, o qual será utilizado para testes.

Atenção: a conformidade com os critérios aqui estabelecidos faz parte da avaliação. Se você não entregar o trabalho no prazo, ou se o executável não for gerado da forma indicada, você receberá nota zero.

Vocês podem assumir que:

- As linhas de cada arquivo nunca terão mais de um milhão de caracteres.
- Todos os caracteres do arquivo estarão entre os seguintes: ‘\n’, ‘\t’ (Tab), números e letras (de ‘a’ a ‘z’ ou ‘A’ a ‘Z’).
- As quebras de linha serão feitas com ‘\n’, assim como no Linux e **Não** com ‘\r\n’, como no Windows.
- Haverá um único ‘\n’ no final da última linha dos arquivos de entrada. Deverá haver um único ‘\n’ no final do arquivo de saída.

Além disso:

- O trabalho fica para meia noite do dia 27/10/2018 (Sábado).
- Os arquivos temporários, criados durante a ordenação externa, não devem ser apagados. Esses arquivos devem ser criados no mesmo diretório do executável *join*.
- Coloquei no PVAnet exemplos de arquivos de entrada e saída.